

Supporting Management of Sensor Networks through Interactive Visual Analysis



Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

DISSERTATION

zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
von

Dipl.-Inf. Martin Steiger
geboren in Regensburg

Referenten der Arbeit: Prof. Dr. techn. Dieter W. Fellner
Prof. Dr. Jörn Kohlhammer
Technische Universität Darmstadt

Tag der Einreichung: 29. Mai 2015
Tag der mündlichen Prüfung: 14. Juli 2015

Darmstädter Dissertation
2015
D 17

Erklärung zur Dissertation

Hiermit versichere ich die vorliegende Dissertation selbständig nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 15. Juli 2015

Martin Steiger

Dedicated to . . .

Maike, Pauline and Frida

Abstract

With the increasing capabilities of measurement devices and computing machines, the amount of recorded data grows rapidly. It is so high that manual processing is no longer feasible.

The *Visual Analytics* approach is powerful because it combines the strengths of human recognition and vision system with today's computing power. Different, but strongly linked visualizations and views provide unique perspectives on the same data elements. The views are linked using position on the screen as well as color, which also plays a secondary role in indicating the degree of similarity. This enables the human recognition system to identify trends and anomalies in a network of measurement readings. As a result, the data analyst has the ability to approach more complex questions such as: are there *anomalies* in the measurement records? What does the network *usually* look like?

In this work we propose a collection of *Visual Analytics* approaches to support the user in exploratory search and related tasks in graph data sets. One aspect is graph navigation, where we use the information of existing labels to support the user in analyzing with this data set. Another consideration is the preservation of the user's mental map, which is supported by smooth transitions between individual keyframes. The later chapters focus on sensor networks, a type of graph data that additionally contains time series data on a per-node basis; this adds an extra dimension of complexity to the problem space. This thesis contributes several techniques to the scientific community in different domains and we summarize them as follows.

We begin with an approach for network exploration. This forms the basis for subsequent contributions, as it to supports user in the orientation and the navigation in any kind of network structure. This is achieved by providing a showing only a small subset of the data (in other words: a local graph view). The user expresses interest in a certain area by selecting one of more focus nodes that define the visible subgraph. Visual cues in the form of pointing arrows indicate other areas of the graph that could be relevant for the user. Based on this network exploration paradigm, we present a combination of different techniques that stabilize the layout of such local graph views by reducing acting forces. As a result, the movement of nodes in the node-link diagram is reduced, which reduces the mental effort to track changes on the screen. However, up to this point the approach suffers from one of the most prominent shortcomings of force-directed graph layouts. Little changes in the initial setup, force parameters, or graph topology changes have a strong impact on the visual representation of the drawing. When the user explores the network, the set of visible nodes continuously changes and therefore the layout will look different when an area of the graph is visited a second time. This makes it difficult to identify differences or recognize different drawing as equal in terms of topology. We contribute an approach for the deterministic generation of layouts based on pre-computed layout patches that are stitched at runtime. This ensures that even force-directed layouts are deterministic, allowing the analyst to recognize previ-

ously explored areas of the graph. In the next step, we apply these rather general purpose concepts from theory in practical applications.

One of the most important network category is that of sensor networks, a type of graph data structure where every node is annotated with a time series. Such networks exist in the form of electric grids and other supply networks. In the wake of distributed and localized energy generation, the analysis of these networks becomes more and more important. We present and discuss a multi-view and multi-perspective environment for network analysis of sensor networks that integrates different data sources. It is then extended into a visualization environment that enables the analyst to track the automated analysis of the processing pipeline of an expert system. As a result, the user can verify the correctness of the system and intervene where necessary. One key issue with expert systems, which typically operate on manually written rules, is that they can deal with explicit statements. They cannot grasp terms such as “uncommon” or “anomalous”. Unfortunately, this is often what the domain experts are looking for. We therefore modify and extend the system into an integrated analysis system for the detection of similar patterns in space and in different granularities of time. Its purpose is to obtain an overview of a large system and to identify hot spots and other anomalies. The idea here is to use similar colors to indicate similar patterns in the network. For that, it is vital to be able to rely on the mapping of time series patterns to color. The Colormap-Explorer supports the analysis and comparison of different implementations of 2D color maps to find the best fit for the task.

As soon as the domain expert has identified problems in the networks, he or she might want to take countermeasures to improve the network stability. We present an approach that integrates simulation in the process to perform “What-If” analysis based on an underlying simulation framework. Subsequent runs can be compared to quickly identify differences and discover the effect of changes in the network.

The approaches that are presented can be utilized in a large variety of applications and application domains. This enables the domain expert to navigate and explore networks, find key elements such as bridges, and detect spurious trends early.

Deutsche Zusammenfassung

Mit der wachsenden Leistung von Messgeräten und Rechnern wächst die Menge der für eine Datenanalyse zur Verfügung stehende Menge an Rohinformation in einem Maße an, dass ein händisches Durchsuchen und Verarbeiten für eine Analyse unmöglich macht.

Die vorliegende Arbeit setzt auf den *Visual Analytics*-Ansatz, der die Fähigkeiten des Menschen – vor allem die visuelle Wahrnehmung und Verständnis – mit der Rechenleistung heutiger Computer in einem Prozess vereint. Damit lassen sich auch Fragen beantworten, die nur schwer explizit zu formulieren sind: gibt es *ungewöhnliche* Verlaufskurven? Wie sieht das Netz *normalerweise* aus?

In dieser Arbeit werden neue Ansätze und Visualisierungstechniken für die Datenanalyse von Netzwerkdaten vorgestellt. Dabei geht es zunächst um Graphen im allgemeinsten Sinne, später fokussiert sich die Arbeit auf sogenannte *Sensor-Netze*. Das sind Netze, die an jedem Knotenpunkte eine kontinuierliche Variable über die Zeit hinweg messen. Analyselösungen für solche Daten sind in vielen Anwendungsgebieten von ungeheurer Bedeutung, sei es in der Steuerung von Strom- und Wasser- und anderen Versorgungsnetzen, bei der Überwachung von Funknetzen. Der Beitrag dieser Arbeit zur visuellen und interaktiven Analyse von Graphdaten liegt ganz konkret auf den folgenden Aspekten.

Zunächst wird ein Ansatz für die Exploration von Netzwerken vorgestellt. Dieser bildet die Grundlage für die darauffolgenden Beiträge, da er den Nutzer bei der Orientierung und Navigation in in lokalen Ansichten von Netzwerken (*dynamic graph views*) im allgemeinsten Sinne unterstützt. Das wird dadurch erreicht, dass nur ein kleiner Teil des Graphs dargestellt wird. Diese Art der Darstellung wird auch als *lokale Ansicht* bzw. *local graph view* bezeichnet. Der Benutzer des Systems drückt sein Interesse an einem Bereich des Graphen aus, indem er oder sie eine oder mehrere Fokusknoten auswählt und dadurch sichtbaren Bereich definiert.

Visuelle Hinweise, die die Form von Richtungspfeilen haben, zeigen dem Benutzer Wege zu anderen Teilen, die relevant sein könnten. Aufbauend auf diesem Paradigma zur Exploration von Netzwerken, wird eine Kombination von verschiedenen Techniken vorgestellt, die das Layout solcher lokaler Ansichten stabilisieren. Deren Kernziel dabei ist die wirkenden Kräfte auf ein Minimum zu reduzieren, um die Bewegung der Knoten im Layout zu reduzieren. Das wiederum macht es leichter, den Veränderungen auf dem Bildschirm zu folgen. Bis zu diesem Punkt jedoch leidet das Layout des Netzwerkes unter einem der markantesten Nachteile von Kräftebasierten (*force-directed*) Layouts. Selbst kleine Änderungen bei der initialen Platzierung der Knoten, in der Konfiguration der Parameter oder Graph-Topologie haben einen starken Einfluss auf die Zeichnung.

Wenn der Benutzer das Netzwerk erkundet, dann ändert sich die Menge der sichtbaren Knoten ständig. Daher sieht das Layout daher jedes Mal anders aus, wenn ein bestimmter Teilbereich erkundet wird. Das macht es schwierig, topologische Unterschiede zu erkennen oder eine bereits

besuchte Region als solche wiederzuerkennen. Um dem zu begegnen, stellen wir einen Ansatz für die deterministische Erzeugung von Layouts vor. Das wird durch Vorberechnen von Layouts basierend auf vielen kleineren Teilen des Graphen ermöglicht. Diese, zur Laufzeit fixen Layouts, werden bei Bedarf entsprechend zusammengefügt, so dass ein kohäsiver Gesamtgraph entsteht. Der Ansatz stellt sicher, dass selbst Kräfte-basierte Layouts deterministisch erzeugt werden können.

Im nächsten Schritt werden diese eher allgemein gehaltenen Konzepte von der Theorie in die Praxis übertragen. Eine der interessantesten Anwendungsgebiete sind Sensor-Netzwerke, ein Typ von Graph, der jedem Knoten eine Zeitserie zuordnet. Beispiele für solche Netzwerke sind Stromnetze und andere Versorgungssysteme. Durch die zunehmende Verkleinerung und Verteilung der Energieerzeugung wird die Analyse des Stromnetzes zunehmend wichtiger. Wir stellen eine Analyseplattform für die professionelle Datenanalyse vor, die verschiedene Ansichten auf die Daten kombiniert. Unterschiedliche Perspektiven ermöglichen das Umschalten zwischen unterschiedlichen Arbeitsabläufen. Verschiedene Datenquellen können am Beispiel der Analyse von Stromnetzen integriert und kombiniert werden.

Dieses System wird dann um eine Komponente erweitert, die automatische Analyse durch ein Expertensystem ermöglicht. Dem Benutzer wird es aber ermöglicht, die automatische Analysepipeline zu überwachen und, wenn nötig, einzugreifen. So kann das Verhalten überprüft, verifiziert und wenn nötig auch eingegriffen werden. Eines der Hauptprobleme von Expertensystemen ist die Abhängigkeit zu den üblicherweise händisch angelegten Regelwerken, die jegliches Verhalten explizit darlegen müssen. Das macht es unmöglich, Terme wie „ungewöhnlich“ oder „auffällig“ zu verwenden, was aber notwendig wäre, da manche Situationen nicht genauer definiert werden können.

Daher wird das System modifiziert und erweitert, so dass es dem Analysten ermöglicht wird, ähnliche Muster und Verläufe im Raum und über die Zeit (in verschiedenen Granularitäten) zu erkennen. Das System gibt einen informativen Überblick über das Gesamtsystem und erlaubt es, schnell und präzise kritische Stellen und Anomalien zu identifizieren. Dem Attribut *Farbe* kommt dabei eine Schlüsselrolle zu, da sie Ähnlichkeit in Mustern widerspiegelt. Daher ist es unbedingt notwendig, dass die Kodierung von Ähnlichkeit in zeitlichen Verläufen korrekt in Farbähnlichkeit abgebildet wird. Der Colormap-Explorer, der im Anschluss vorgestellt wird, unterstützt die Analyse und den Vergleich von unterschiedlichen Farbkarten (*colormaps*), um den geeignetsten Kandidaten zu finden.

Sobald der Domänenexperte Probleme im Netzwerk identifiziert hat, können dann mögliche Gegenmaßnahmen evaluiert werden. Wir stellen einen Ansatz vor, der Simulation in den Prozess integriert, um eine Sensitivitätsanalyse durchführen zu können. Verschiedene Durchläufe der Simulation können damit schnell und effektiv auf Knotenbasis verglichen werden, um die Auswirkung von geplanten Änderungen leichter abschätzen zu können. Das ermöglicht dem Planer genauere Vorhersagen von Folgeeffekten durch Veränderungen im Netz zu machen.

Die vorgestellten Techniken können für eine Vielzahl von Anwendungen und Anwendungsgebieten realisiert werden. Dies führt dazu, dass die Domänenexperten die Möglichkeiten besitzen, schnell und sicher in Netzwerken zu navigieren, Schlüsselstellen zu finden und Anomalien in den Daten frühzeitig zu erkennen.

Contents

1. Introduction	1
1.1. Motivation	1
1.2. Combining Strengths of Man and Machine	2
1.3. Contribution	3
1.4. Thesis Outline	4
2. Fundamentals of Information Visualization	7
2.1. History of Information Visualization	7
2.2. Tasks in Information Visualization	8
2.3. Information Visualization Models	10
2.4. Data Type Taxonomy and Visualizations	12
2.4.1. One-dimensional and Temporal Data	13
2.4.2. Two- and Three-dimensional Data	13
2.4.3. Multi-dimensional Data	14
2.4.4. Hierarchies and Trees	19
2.5. Visual Interaction	21
2.5.1. Direct Manipulation	21
2.5.2. Brushing and Linking	21
2.5.3. Panning and Zooming	22
2.5.4. Animation	23
2.5.5. Focus plus Context	23
2.5.6. Overview plus Detail	24
2.6. Visual Analytics	24
2.6.1. Related Research Fields	25
2.6.2. The Visual Analytics Process	25
2.7. Summary	27
3. Local Graph Views	29
3.1. Background	30
3.2. Requirement Definition	31
3.3. Related Work	34
3.3.1. Adjacency Matrix Displays	34
3.3.2. Node-link Diagrams	35
3.3.3. Force-directed Layouts	36
3.3.4. Projection-based Layouts	37

3.3.5.	Hierarchical Graph Layouts	37
3.3.6.	Dynamic Graphs	38
3.3.7.	Orientation & Navigation	39
3.3.8.	Preserving the Mental Map	39
3.3.9.	Degree of Interest	40
3.3.10.	Off-Screen Visualizations	41
3.4.	Definition of Graph, Context and Focal Node	42
3.4.1.	Initial Node Selection	42
3.4.2.	The Degree-of-Interest Function	43
3.5.	Signposts	43
3.5.1.	Visual Cues	44
3.5.2.	Defining the Local View	45
3.5.3.	Signpost-based Context	49
3.5.4.	User Study	54
3.5.5.	Discussion	56
3.6.	Stabilized Layouts	57
3.6.1.	Damping Node Movement	57
3.6.2.	Placing Nodes	58
3.6.3.	Test Results	62
3.6.4.	Discussion	67
3.7.	Deterministic Shape Matching	67
3.7.1.	Background and Overview	67
3.7.2.	Concept	69
3.7.3.	Preliminary Tests	74
3.7.4.	Discussion	78
3.8.	Deterministic High-dimensional Layout Stitching	79
3.8.1.	Concept	79
3.8.2.	Proof of Concept	82
3.8.3.	Discussion	83
3.9.	Summary	83
4.	Visual Analysis of Sensor Networks	87
4.1.	Background	88
4.2.	Requirements	89
4.3.	Sensor Network Analysis	92
4.3.1.	Automated Event Analysis	92
4.3.2.	Visualization Systems	94
4.4.	Network Layout Strategies	96
4.4.1.	Types of Network Representation	96
4.4.2.	Design Space Criteria	98
4.4.3.	Layout Algorithms	99
4.4.4.	Visual Interaction Concepts	103
4.4.5.	Discussion	106

4.5. Sensor Network Monitoring through Visual Analysis	109
4.5.1. Combining Automatic and Manual Analysis	109
4.5.2. Domain Model and Event Processing	111
4.5.3. A Rule-based Analysis Pipeline	112
4.5.4. Visualization & Interaction	113
4.5.5. Discussion	117
4.6. Anomaly Detection in Sensor Networks	119
4.6.1. Revealing Similarity of Time-series Data	120
4.6.2. Data and Algorithms of the System	121
4.6.3. Visualization & Interaction	124
4.6.4. Case Study	131
4.6.5. Design Process	132
4.6.6. Discussion	133
4.7. Explorative Analysis of 2D Color Maps	133
4.7.1. Visual Access to Color Maps	134
4.7.2. Encoding Information in Color Maps	136
4.7.3. Perceived Color Differences	137
4.7.4. The ColorMap-Explorer	138
4.7.5. Discussion	150
4.8. Exploring Simulation in Sensor Network Models	150
4.8.1. Visual Support for Simulations	150
4.8.2. Concept	154
4.8.3. Case Study	159
4.8.4. Discussion	161
4.9. Summary	162
5. Conclusions and Outlook	165
5.1. Future Work	166
5.1.1. Local Graph Views	167
5.1.2. Sensor Networks	167
5.2. Outlook	168
A. Publications	171
B. Supervising Activities	175
Bibliography	177

1. Introduction

1.1. Motivation

The past years have seen a strong growth of all types of data that require analysis. This development is increasingly driven by the machine-based generation of data. Prominent examples are the automated logging and recording of telecommunication connection and web access, short-range tag tracker such as RFID sensors. Large amounts of data are recorded in many application domains; this includes transaction data in finance, energy consumption of individual customers of electricity providers and environmental information in meteorology. With the increased linking of data sets, the connectivity as such turns into a potential source of information and insight. Thus, analysis of network data has gained strong interest because, in contrast to tabular data, explicit links between data inherent to the network system exist. One of the most important types of networks are sensor networks. These networks consist of nodes that measure one or more continuous variables over time.

One example is the Stream Gauge Networks that most industrialized countries have set up to monitor rivers and other water bodies. Gauging stations are located alongside rivers, forming partially connected networks at river joins. In these stations, quantities such as the surface water level (the *stage*) and the flow of the water are continuously measured. Aside from water quality assessment, geo-scientists use these networks to study hydrologic extremes such as floods and droughts.

Another example is the measurement of energy consumption of households. Electricity is brought to the customers through a network of electric cables and transformer stations. These transformer stations also measure the flow of electric current, allowing for drawing conclusions on consumption patterns of individual consumers or groups of them. This is useful for control room operators in managing the production, distribution and transport of energy.

The amount of sensor-based data has seen a rapid growth over the past years across many different disciplines and applications. With more and more sensors being installed, the higher resolution of measurement devices, the recorded data grows exponentially and becomes overwhelming. Apart from dealing with these extremely large amounts of data per se, a major challenge is to extract that information that is most relevant for the analysis from these data sets. This information could be trends, repeating patterns, spikes, etc. and depends strongly on the application context. The relevance of sensor networks in many practical applications motivated us to investigate the problem domain in detail.

1.2. Combining Strengths of Man and Machine

The problem of analyzing sensor network and network data in general is both large and complex. High resolutions of sensors combined with the large number of installed recording devices leads to super-linear growth of the data bases. On top of that, the partially unknown reliability and the increasing heterogeneity of the individual data sources drive the complexity.

To tackle the size aspect of the problem, machines are necessary. Today, even mobile devices are able to compute millions of operations every second. Commodity hardware even allows billions, as well as the ability to process (analyze) large quantities of data on multiple processors. This computing power can be exploited in automated algorithms to find solutions to certain problems in little time with little user involvement.

An inherent requirement, however, is that the goal must be made explicit (e.g. “find users in a database that are between 30 and 60 years old”) and the instruction algorithm for finding a solution must be clearly laid out for machine processing. When this requirement is fulfilled, the computer can extract the relevant information very precisely. An algorithmic approach can find the best and fastest solutions (with respect to certain constraints, limitations and quality criteria) faster than any human. In such cases, other means such as visualization are then not required to find such a solution [vW05].

The process of extracting information per se is trivial in such explicit settings: the data contains information. This information is foraged by an algorithm and then communicated to the user that seeks to gain insight in the data. A common issue is that users must assume the algorithm (as well as the resulting data analysis) to be correct. According to van Wijk, such automated approaches need to be “fool-proof” [vW05] in order to be usable.

Unfortunately, the situation is far less clear in many practical applications. Hydrologists want to see signs of an upcoming flood in a network of gauging stations. Energy providers want to obtain an impression of consumption patterns of their customers. Generally speaking, data analysts want to get a “feeling” (often referred to as *insight*) of the information in a data set. This is often difficult to convey explicitly to machines, so that they will yield the information desired by the user. What does a typical pattern look like? What is an anomaly?

How do we tell computers how this goal can be reached? One approach is *Machine learning*, a group of algorithms and techniques that construct models based on input data. However, neither the problem nor the expected results of the process can be explicitly formulated. Such tasks are often described as exploratory search in the uncharted region of a data set. This is true for tabular data and even more so for network data, which also defines relations between individual items.

The most prominent claim of *Information Visualization* is that its purpose is the aforementioned *Insight* [vW05, Nor06]. Thomas and Cook propose the integration of the user through visualization in the process to take advantage of certain human abilities [TC05]. These are, for example, the human vision and the perceptual system. Humans are able to identify visual structures and patterns in images faster and at unrivaled accuracy and precision. Computers on the other hand perform millions of mathematical operations in a single second. The aim of interactive data analysis is to extract or generate insight from data sets by combining the advantages of human visual recognition with the strengths of machine-based computation. This is discussed in more detail in Section 2.6.

1.3. Contribution

One of the most central problems domain experts are currently confronted with is the very abstract question about what information is hidden in the data they hold in their hands. Generally speaking, this is independent from the data type per se. This question is particularly interesting for graph data since it also involves connectivity between data items, but also challenging. Colloquially spoken, domain experts such as control room operators need to know what is going on in their networks. Only on the basis of precise knowledge of the situation – in both temporal and spatial domains – they are able to make sound decisions. This thesis approaches this problem with a set of techniques to enable the analyst to gain insight in the details of large networks and find important connections between different parts. On this basis, the analyst can find answers to practical questions such as: which parts of the networks behave similarly? Which ones are affected by seasonal effects? What is the expected effect of changes in the topology?

In this work, the aforementioned combination and intertwining of computer and human strengths is used as a key component for the exploratory analysis of data sets. We propose a collection of approaches to support the user in exploratory search and related tasks in graph data sets. One aspect is graph navigation, where we use the information of existing labels to support the user in working with this data set. Another one is the preservation of the mental map which is supported by smooth transitions between individual keyframes. Deterministic layout enables the user to recognize previously explored parts of a graph.

While these contributions work with any type of graph data, we also claim to contribute to a specific type of graph data, namely sensor networks. This adds an extra dimension of complexity to the problem space. It involves not only also dealing with a different data type (time), but requires dealing with time series data for every node of the graph. We approach this problem with a combination of different visualizations and views and are strongly linked and provide different perspectives on the same data elements. Aside from the position on the screen, color plays a key role, as it is used to link different views on the same data elements, but also to indicate similarity between different data elements. This enables the human recognition system to identify trends and anomalies in a network of measurement readings. We claim to contribute several items to the scientific community in different domains and summarize them as follows:

- The baseline of this work forms an approach for network exploration using local graph views that support multiple focus nodes. It also adds visual cues to support the user's orientation and navigation in the most common sense. This contribution C_1 was published in [MSDK12].
- Based on that, we present an approach that stabilizes the layout of local graph views by reducing acting forces with a combination of four different techniques. The aim is to reduce the cognitive effort of the user that is exploring the network by damping node movement. This contribution C_2 was first published in [SMK13] and in an extended version as a journal article in [SMK14].
- However, small changes in the data or parametrization still have a strong impact on the display. We counter this with pre-computed layout patches that are stitched at runtime.

This ensures that even such force-directed layouts become deterministic. Thus, the analyst can recognize previously explored areas of the graph. This contribution C_3 was published in [SLTM*13] and an extension for high-dimensional layouts in [SLTM*14].

- Taking these approaches from theory into practice, we present a multi-view and multi-perspective environment for network analysis of sensor networks. This includes recorded time series data, but also integrates different data sources. This was done in the application domain of electric grids and published in [SMDK13a] (Contribution C_4).
- This visualization environment was enhanced by integrating automated analysis to also deal with large data sets. It enables the analyst to track this automated analysis of the processing pipeline. As a result, the user can verify the correctness of the system and intervene where necessary. This contribution C_5 was published in [SMDK13b].
- Unfortunately, expert system require explicit statements, which makes is hard to identify anomalies. We present an extension to the system that allows for the detection of similar patterns in space and in different granularities of time. Its purpose is to get an overview over large network systems and to identify hot spots and other anomalies. This contribution C_6 was published in [SBM*14].
- Mapping time series data to color depends strongly on the choice of color maps. We present the ColorMap Explorer, a tool for the analysis and comparison of such maps based on quantitative and qualitative criteria. It comes with a set of over twenty color maps from scientific literature that visualization designers can use (Contribution C_7).
- Finally, when the domain expert has identified deficits in the network structure, different countermeasures can be analyzed with a visualization system we present. It integrates simulation in the process to perform “What-If” analysis based on an underlying simulation system. Different runs can be compared to quickly identify differences. This contribution C_8 was performed on the basis of water networks and published in [SHS*14].

The contributions are explained in detail in the following chapters of the thesis.

1.4. Thesis Outline

Following this introductory chapter, an overview on visualizations fundamentals is given. It covers a brief history of Information Visualization in general, then outlines tasks and task models that have been identified by the scientific community (Chapter 2). Two complementing models for information visualization are then presented. One is by Card et al. who focus on data transformation, the other one by Liu and Stasko which explains the human internalization process of visualizations. This is followed by a data type taxonomy along with a few visualization examples on how these data types are typically displayed. The most relevant data type – graph data – is discussed later in a separate chapter. Visualization techniques are usually complemented by interaction techniques, to integrate the user in the analysis. Then, the most important interaction

types are described. In the last section Visual Analytics is presented as a research field that tightly combines the strengths of human and machine in an iterative process.

Based on these fundamentals, Chapter 3 (Local Graph Views) explicitly discusses problems that arise when *general* graph and network data are analyzed in local views. Several important requirements for graph analysis are defined, followed by an introduction to the topic. Previous approaches are discussed before a formal definition of all discussed items is given. The main part of this chapter is a set of contributions to the scientific community. Starting with an approach to define local graph views on the basis of multiple focus nodes, visual cues in the form of colored arrows are attached to support orientation and navigation tasks (C_1). Based on that, several techniques are presented to reduce the mental effort in the navigation process ($C_{2,3}$). Key goals are the reduction of node movement and the generation of a deterministic layout so that the user can recognize previously explored areas of the graph. The individual contributions for this chapter are derived directly from the previous section. In the conclusion a comparison between the requirements for a successful analysis and the claimed contributions is made. A summary marks the end of this chapter of the thesis.

In Chapter 4 (Visual Analysis of Sensor Networks), the theoretical contributions are enhanced and applied in practical scenarios. This is performed on the basis of sensor networks, a type of network that is highly relevant to many real-world applications. They are special in that they also record data over time in each of the network nodes. The structure of the chapter is analogous to the structure of the previous chapter. After a short introduction and dissociation from the general graph analysis part, requirements are laid out. Many real-world sensor networks are referenced by geographical coordinates in one way or another; these can be utilized within the layout strategy. A survey of existing strategies and recommendations is given, before a series of contributions to the scientific community is presented and discussed. An analysis system for the visual analysis of sensor networks is presented and extended in several iterations ($C_{4,5,6}$). It integrates different data sources, combines automated processing through expert system with the human ability to identify trends and patterns in the data. As a result, the domain expert is able to find weaknesses or anomalies early on. One key element for the analysis of temporal pattern is a selected 2D color map. With the help of ColorMap-Explorer, a tool for the interactive analysis and comparison of different color maps, the best-fitting implementation can be identified (C_7). In the last step, different countermeasures based on a simulation forecast can be evaluate their effect on the network as a whole and per node (C_8). A conclusive chapter at the end compares requirements and claims before the content of the chapter is summarized.

Chapter 5 concludes this thesis. It summarizes its main challenges and contributions and gives an outlook for future developments in the related fields on a broad scale.

2. Fundamentals of Information Visualization

In this section, we will discuss the fundamentals of information visualization. After a short introduction based on the most prominent examples of information visualizations from history, different tasks and processing models are discussed. Then, the most relevant data types are introduced, followed by different visualization categories, each with one or more examples. Categories that are relevant for the later chapters are discussed in more detail than the others. An overview on interaction techniques completes the overview on interactive visual analysis. Finally, Visual Analytics is introduced as approach for exploratory analysis.

The content of this chapter has been partially published in the book chapter “Information Visualization and Policy Modeling” [NSBK14].

2.1. History of Information Visualization

The scientific domain of information visualization is derived from several different sources like scientific visualization and human-computer interaction [Sii07]. The foundation of the domain was created in 1786 by William Playfair who brought the bar chart, pie chart and several other diagrams to life [SW97]. He is thus often referenced as the founder of graphical methods of statistics. The British nurse Florence Nightingale illustrated seasonal influences on the mortality rate in a military field hospital using polar area charts for the first time in 1857 [Bos08]. She was rewarded as the first female member of the Royal Statistical Society in 1859 for her contributions in statistical graphics. One of the most renowned diagrams of the 19th century is the flow visualization (a variant of a Sankey-diagram) of Napoleon’s campaign against Russia created by Charles Minard in 1869.

The English draftsman Harry Beck created the first topological diagram in 1931 which was used as a map for the London underground [GB94]. The average traveler is interested in getting transported from one station to another. The topology of the railway matters more to him than the physical locations of the stations. So Beck developed a simplified map of the stations that was no longer a geological map. The connecting lines run only vertically, horizontally or on 45 degree diagonals. Many contemporary tube maps are based on Beck’s design. We will refer to this conceptual work in Section 4.4. The cartographer Jacques Bertin, who is today mostly renowned for his 1967 book “Semiology of Graphics” worked on the theory of graphical representations with a focus on cartography [Ber67]. One remarkable result of his work was the reorderable matrix, the first interactive visualization method for multidimensional data.

Due to advances in computers and computer graphics in particular it was now possible to create completely new visualization techniques based on pixel-based graphics. An important image-distortion technique, the fisheye view [Fur86], was presented. The group of Card, Robertson and

The second task (Comparison) is about comparing values in one attribute or between different attributes. In the Integration task, information from different attributes is combined.

Similarly, Keim et al. define three main roles for visualization in the context of data analysis: result presentation, confirmatory analysis and exploratory analysis [KMS*08]. The first of the three roles is about communicating previously identified findings to an audience. In confirmatory analysis, the user already has a hypothesis in mind that should be refuted or confirmed, before the analysis process is started. In contrast to that, exploratory analysis is about gaining insight in an data set. No a-priori knowledge is available, nor any explicitly stated aim of the analysis process. During the process, new hypotheses can be formulated. In this work we neglect the result presentation part in favor of analytical visualization, in particular its explorative aspect.

Brehmer and Munzer discuss a hierarchical typology for visualization tasks [BM13]. The authors criticize previous work on task taxonomies for not acknowledging the user and task environment enough. They therefore first categorize tasks by the user's motivation: *why* is a task performed? At the next level, the question on *how* the task is performed before asking about the task's input and output variables (the *what?* aspect).

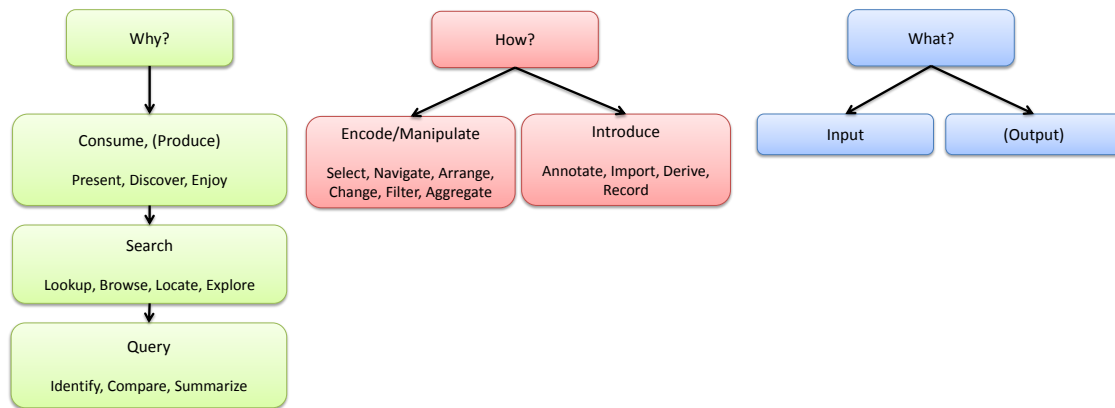


Figure 2.2: An adapted and simplified version of the multi-level task typology by Brehmer and Munzner [BM13]. It arranges tasks by asking three questions: *why?*, *how?* and *what?*.

In agreement with Keim, Brehmer and Munzer see presentation and discovery (confirmatory and exploratory) as major drivers for visualization. They define it as *consumption*, because the user wants to consume the information that is hidden in the data. This is followed by the *search* process where the user tries to find interesting items. In some cases, the location may be unknown, in some the actual target is unknown and in some cases even both. Lastly, the user performs a *query*, for example to identify specific elements or compare different attributes. In the second part of the typology, Brehmer and Munzner define general methods (visual encodings and interaction techniques) to achieve the targeted transfer of information. This is depicted in the center column in Figure 2.2. Three different categories are defined: *encode* information as a visual representation, *manipulate* existing visual elements (often through interaction) and *introduce* new elements such as annotations. The *what* part of the model discusses the *input* and – if applicable – the *output* of

visual interfaces. For example, input data can be tabular or graph data sets, sometimes specified on a per-attribute basis (ordinal vs. categorical).

This typology allows data analysts to break down complex tasks in simpler ones and to explicitly specify connections between them. Highly related to this task-based classification are models for the information visualization process. We will highlight the most relevant ones for this work in the next section.

2.3. Information Visualization Models

One of the most influential technical model in information visualization is the model of Card, Mackinlay and Shneiderman (see Figure 2.3). It is a data flow diagram that models the data processing from its raw form into a visual representation. The visualization is described as a series of partly independent transformations. Its main contribution is that the complexity of the visualization process is split into smaller sub-processes. This is why it still serves as a basis for many visualization system architectures today. Usually, scientific contributions in the information visualization domain can be mapped precisely onto particular parts of the pipeline. Another important aspect of their work is the idea of user interaction in the pipeline. A visualization technique is not static process. Every component along the data processing pipeline serves as a basis for process control mechanisms.

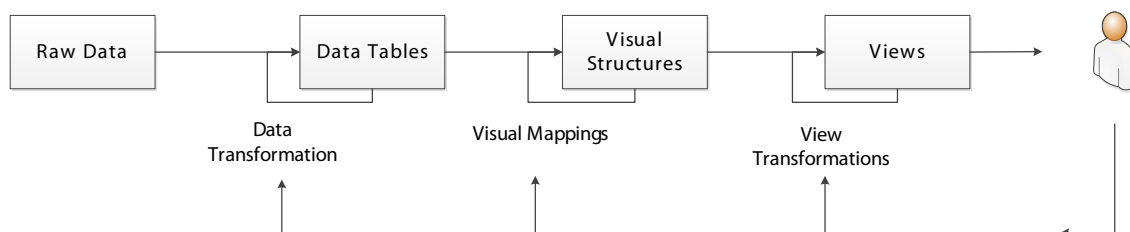


Figure 2.3: Data Flow Pipeline in Information Visualization as described by Card, Mackinlay and Shneiderman (Own drawing).

The pipeline begins with the transformation of the raw input data into data formats that are suitable for the visualization. This standardization is necessary if more than one data source should be attached to the process or if a single data source is used for different visualization techniques. The transformation aims at a data representation that is normalized in terms of content and structure so that the visualization can be decoupled from the input data. This is an important strategy that permits adapting techniques to different scenarios and data sets. It might involve trivial operations like converting one data format into another, but in many cases it is also necessary to identify and deal with incomplete, imprecise or erroneous data. Depending on the application the outcome of this step is well-defined data for the visualization.

The second transformation element in Card's visualization pipeline is the mapping of standardized, but raw data items into the visual space. This mapping can be considered as the core transformation that forms the actual visualization. That is why the different visualization tech-

niques can be differentiated in this part of the pipeline. The visual space is described by a series of visual attributes which inherently represent the basic tools of the visualization techniques. Ware identified four groups of these attributes: form, color, animation and space [War12].

While the second part of the pipeline describes the transformation into the visual space, the third element is about transformations within the visual space: the view transformation. In almost any case the transformation also takes place within the value set of a single visual attribute. This includes, for example, rotation, zoom and other camera settings as well as modifications of the color map for an attribute. Card's model of the visualization pipeline is also a model for a technical realization of visualization techniques and processes.

In many cases, the approaches for the theory and the models in information visualization can be assigned to one of two groups. These are “data-centered” and “decision- or user-centered” tasks. They differ mainly by the information that is available in the design phase. Amar and Stasko put those two principles in juxtaposition in the context of information visualization [AS05].

Visualization in data-centered approaches aims at a realistic representation of data and its structure. In its most consequent form, this idea is completely independent of the human user and the tasks that should be solved using that visualization. Its main goal is to create an identical replication of the input data in the mental model of the user. Viewing the data is an elementary low-level process. It is supported through visualization, but it does not support the user in solving a high-level task. According to Amar and Stasko, the static connection between analytic activities is based on the assumption that the aims of the user are also formulated in a static and explicit manner. They find it necessary to link the user tasks on different abstraction layers through information visualization, i.e. low-level and high-level tasks.

More recently, Liu and Stasko investigate the analytical process from the perspective of different user groups [LS10]. How does the mental model in the analyst's mind relate to the visualization that is visible to the eye? The authors define the expression *mental model* as a functional representation of an interactive visualization system. Notably, they also draw a line between mental model and mental map. The authors define a mental model to be more abstract and represents data on a very high level, but in contrast to a map, it must be *functional*. The functional aspect is critical for the reasoning process which the authors define as “[..] constructing and simulating a mental model”. We agree with the authors and summarize the definition of a mental map relates more to spatial organization. This definition is also relevant for Section 3.3.8, where publications with a focus on the preservation of this mental map are discussed. In contrast to the mental model, a mental map preserves all relevant properties in terms of structure and behavior and sometimes also specific information about the analyzed data set without spatial reference. However, the user is able to reason with this internal representation.

The model describes the interaction between the *external* visualization and the internal *mental model* in four discrete steps. This process is illustrated in Figure 2.4.

1. *internalize*: Liu and Stasko refer to this process of converting the visualization to such a mental model as internalization. It can be seen as a learning process that – among others – depends on human factors such as user experience with visualization systems. Getting to know the structure and semantics of the data behind the visualization is also an important part of this process.

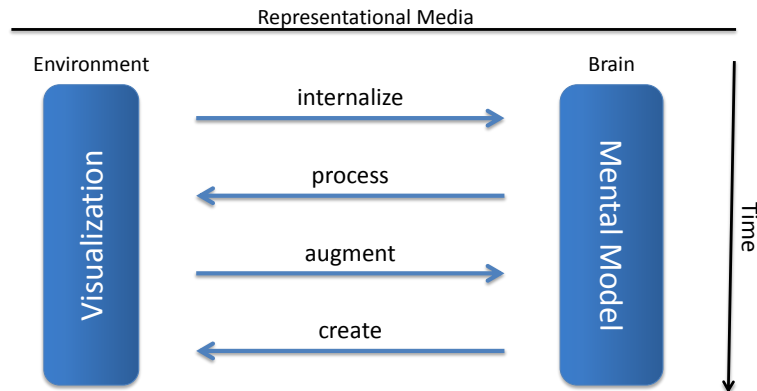


Figure 2.4: The internalization process as defined by Liu and Stasko [LS10] (Own drawing). Through representational media, visualization is perceived in the human brain. The mental model is refined in an iterative, interactive process (Own drawing).

2. *process*: As soon as the model is learned, it can be used to analyze new views. The human perception system tries to make sense of previously unknown views based on the mental model of known visualizations. However, this process has not studied in depth and therefore is not yet fully understood.
3. *augment*: Using the mental model alone for reasoning is often challenging. External visualizations can support this task and improve the existing mental model.
4. *create*: In the last part of the pipeline, the user is enabled to design new visualizations based on the internal mental model. The concept of *analogies* plays a major role in this step. Some ideas are popular mainly because the audience is already familiar with a similar idea.

The steps in this model are not strictly separated and often performed in parallel by the human cognition system. It explains in a generic way the interaction process between a human user and a visualization system.

2.4. Data Type Taxonomy and Visualizations

In the following, we will present two parts of Card's transformation pipeline: the visual mappings and the interaction techniques. Mappings can be partitioned into several partitions that map fundamentally different structures into the visual space. Interaction techniques can be roughly classified by the part of the visualization pipeline they control. In this manner, the differentiation is performed through technical criteria. However, it would also be possible to separate the visualizations by the task they support. Card's information visualization model that was described in the previous section starts with the transformation of data in their raw form. Heterogeneous

data types need to be investigated for the transformation process. Shneiderman introduced a taxonomy of data types, which distinguishes data types in one-, two- and three-dimensional data, temporal and multidimensional data, and tree and network data [Shn96]. We shed light on these categories in this section of the chapter and briefly sketch some of the most prominent visualization approaches for each one of them. Together with an independent taxonomy of analysis tasks, Shneiderman also mentions different visualization techniques that provide solutions for specific tasks and data. It has to be stated, however, that it is quite common that a given data set falls into more than one of these categories of the taxonomy. The term “dimensionality” may either refer to the dimension of the actual data, or to the dimension of the display. In some cases, if the data set has a “intrinsic” dimensionality (as is the case with most geo-spatial data sets) the preferred visualization techniques naturally map this data onto this space. The work of Keim gives a survey on the basis of Shneiderman taxonomies [Kei00].

2.4.1. One-dimensional and Temporal Data

A table with two columns can be seen as a mapping routine that transforms values from one category to another. This is a typical example of a one-dimensional data set. If they contain at least one temporal component in their structure, they are referred to as temporal data set or time series data and form a special subclass of one-dimensional data. This data type can be defined as an ordered list of time-value pairs. Given the usual complexity of input data sets, they do not fall in the category of one-dimensional data alone. In this paragraph we present a number of visualization approaches which emphasize the temporal or one-dimensional components of the data sets. Havre presents a visualization technique called *ThemeRiver* as part of a document analysis of news reports [HHN00]. It maps the change of headline stories in the news onto a time scale. The basis of this technique is the appearance of a specific keyword appearing in a number of articles and shows how specific themes may appear at the same time. Card et al. describe a visualization that maps the temporal data is also onto a single axis, a time-line [CSP*06] in a hierarchical manner. The most prominent approaches for mapping temporal data to a visual element are animation and time-lines. Highly similar to the latter are small multiples, i.e. snapshot representations at certain points in time that are plotted side-by-side. In most cases, one of these variants is chosen, because they can be intuitively understood.

Hao et al. propose a combination of hierarchical data together with a large time-series data set [HDKS05]. In their application scenario, the time-series entities show intrinsic hierarchical relationships. This technique combines the properties of a tree-map with the ability to show temporal development of stock-market prices. The hierarchical properties of the underlying data are used to match the level of interest and importance in the layout.

2.4.2. Two- and Three-dimensional Data

The mapping of abstract two and three-dimensional data has by far the longest tradition. All kinds of geo-spatial information visualization can be identified as a mapping from data in a two-dimensional space (geographical maps) or three-dimensional space (a virtual model of our physical world). Embedding abstract information in a geographic representation is one of the most

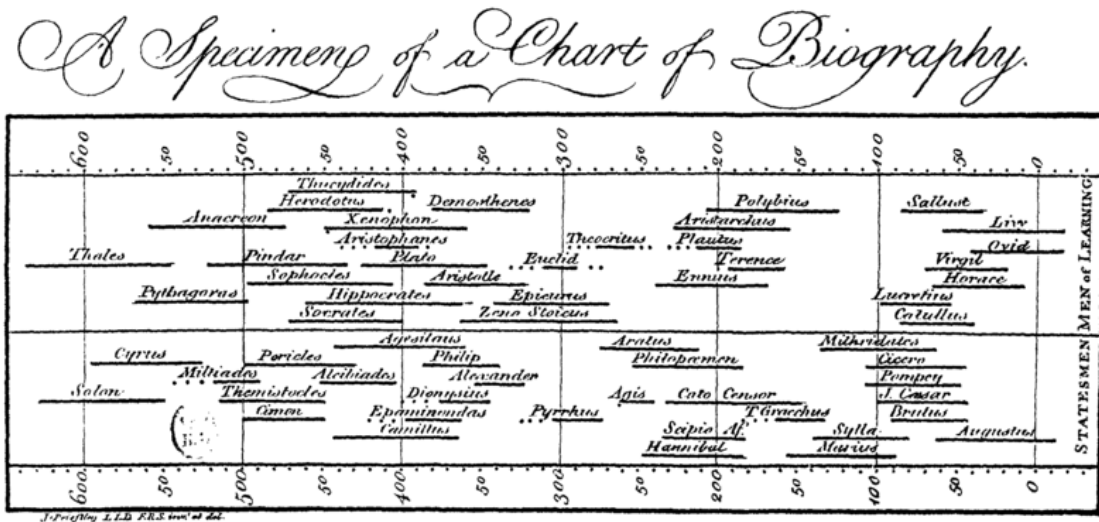


Figure 2.5: Priestley's Chart of Biography (1765), a list of celebrities that is ordered chronologically. The lifespan of an individual is represented by a straight line along the axis. This image is in the public domain. Source: <http://commons.wikimedia.org/wiki/File:PriestleyChart.gif>.

abundant metaphors possible. This is because the reference to a location is one of the most important relations people use to organize information. Hence, many visualization techniques for this embedding have been developed. Over the years, this concept evolved from plain satellite image visualization to sophisticated (collaborative) platforms. The abstract representation of the world serves as a common frame of reference to contribute, search and analyze large amounts of additional geographic metadata. Not surprisingly, many visualization techniques have been developed that use this as a basis for their data [DMK05].

2.4.3. Multi-dimensional Data

One of the most prominent mappings of abstract data into two-dimensional space is the scatterplot technique, which appears in numerous variations [NS00]. They are often used in a grid-based layout (also known as scatterplot matrices) for multidimensional data analysis. Scatterplots work best for numerical data (which can be mapped on the x and y coordinates respectively), and is of limited usage to convey categorical or even purely semantic information. The idea is rather simple: points in n-dimensional space become points in 2-dimensional space. This is why they are often used to visualize projections between n-dimensional data-space and display-space.

Asimov proposed to visualize high-dimensional data sets as a series of animated 2D projections, called *Grand Tour* [Asi85]. However, with increasing number of attributes, the number of possible x/y combinations becomes too large to be visualized in reasonable time. Using *Projection Pursuit*, this number can be reduced based on some measure of interestingness [Hub85]. In "Rolling the dice", Elmqvist proposes a combination of scatterplot matrix for overview and animated 3D

transitions for interactive analysis based on scatterplots [EDF08]. The matrix display provides an overview and helps the user finding interesting attribute combinations.

Visualization techniques for multi-dimensional (sometimes also referred to as multi-variate) data explicitly address the problem to visualize and identify inherent dependencies in the data sets, which cannot be expressed by simple correlations. Such relations may involve many variables. One of the major goals in all of these techniques is to display a sufficiently represent this information in 2-dimensional screen space to make these correlations visible.

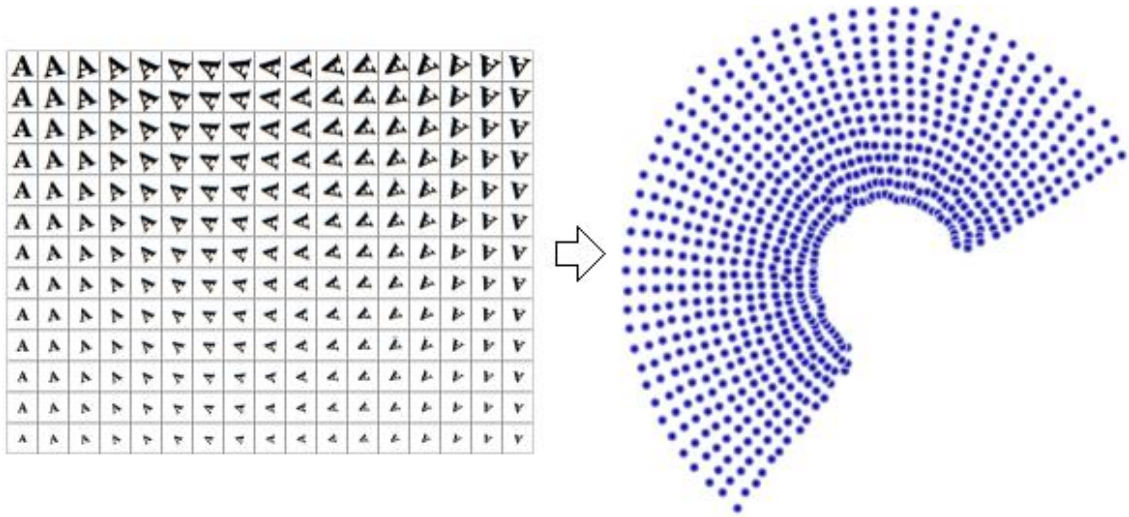


Figure 2.6: Each of the 32x32 pixel letter images is interpreted as a single, binary coordinate in a 1024-dimensional space. However, the intrinsic dimensionality is only two (rotation and scale). Non-linear projection algorithms are able to reduce this dimensionality while preserving most of the contained information. This image is in the public domain. Source: <http://en.wikipedia.org/wiki/File:Nldr.jpg>.

As described in the previous section, the visual mapping is most often used to characterize the visualization technique. It is important to note here that visualization techniques rarely contain only a single visual mapping. Recently published approaches are often combinations of elementary techniques. The work of Keim gives subjective ratings of different techniques and approaches [Kei00]. These are typically related to the technique's ability to solve a particular task rather than the type of data they display. We will not discuss the class of pure iconic techniques in detail, because it has lost importance during the past couple of years. In Keim's classification, every visualization technique that maps a data element directly on a visual attribute that is more complex than a single pixel (e.g. lines, glyphs, etc.) belongs to the group of geometric methods. It contains most of the classical diagrams like starplots, pie charts, bar charts, line charts, histograms, etc. as well as geographic maps. It is highly heterogeneous and contains many hybrids that also belong to class of projection methods.

Most of the recent frameworks and techniques derive their improvements from an adequate combination of different basic techniques – in some cases in the same display. This holds true

especially for glyphs, singular symbols for data objects that represent one or more attributes. They also constitute a group of multidimensional visualization techniques, but do not refer to the layout (i.e. the positioning of visual objects in screen space) but on the appearance of objects. Basically every visual object that conveys more information than its position can be considered as a glyph. Today, the results of this domain are reused particularly in glyph-based designs. Aside from graph-based visualizations for networks and hierarchies (see Section 3.3), three classes of techniques evolved over the years to become prominent representatives for the visualization of multi-dimensional data: The first one is the *parallel coordinate plot (PCP)* and depicted in Figure 2.7, the second is about *pixel-oriented* layouts and more recently the category of *projection-based* approaches.

2.4.3.1. Parallel Coordinates

As the name suggests, the parallel coordinate technique has a number of coordinate axes in the display, arranged in a row of parallel lines. Basically this technique can be used for most data types, but it works best for ordinal and numerical data. A point in the multi-dimensional space is drawn as a poly-line connecting the (coordinate-) values on every axis. While the basic idea is relatively old [Ins85], later studies on parallel coordinates emphasize their use for the analysis of data sets [Sii00]. In many cases, this technique is tightly coupled with the generation of dynamic queries and the identification of data clusters [FWR99]. Surprisingly, the first controlled user study on the effectiveness of PCP (compared to scatter plots) with respect to correlation analysis has been conducted only in 2010 [LMvW10].

2.4.3.2. Pixel-oriented Visualizations

A visualization technique belongs to the group of pixel-based methods if the number of used visual attributes comprises only the position and color of a single pixel. Consequently, every pixel represents a data element which permits to display a maximum number of data elements at the same time. The use of “non-data-ink” is reduced to a minimum. This makes pixel-based techniques suitable for the explorative analysis of patterns and other distinctive features. However, pixel-based methods impose two design-problems. The value set of an attribute must be mapped to the range of available colors, but this is a problem that persists in most visualization techniques. The second problem is about arranging the pixels on the screen while preserving their inherent relations. In many cases there is no strict correspondence between the similarity of the data items and their distance. The general idea of these techniques can be found in the works of Keim [KAK95, KK96].

The visualization can be seen as a function that values from high-dimensional space on the 2D screen. The function that maps data elements in the visual space can be seen as the result of an optimization process. Assuming that the data set is ordered, this optimization must ensure that the one-dimensional ordering is kept also in the two-dimensional display. Equally important is the selection of the display area that ensures that the average distance between pixels that belong to the same data set is minimal. The purpose of that is to aid the user in finding relations between different attributes in a data set.

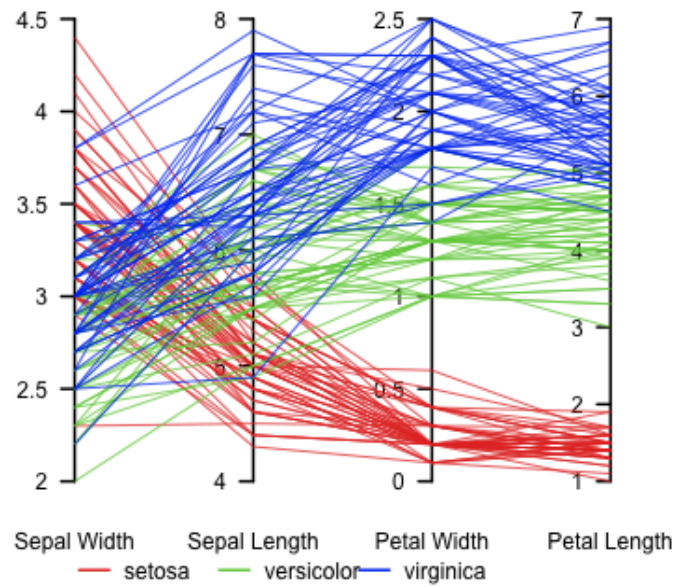


Figure 2.7: A parallel coordinate plot of Fisher's Iris data. The three species are indicated by red, green and blue lines that cross the chart plot at four points to indicate the values of different features. The image is a cropped version of an image that is in the public domain. Source: <http://en.wikipedia.org/wiki/File:ParCorFisherIris.png>

May et al. present a visualization technique that maps multiple attributes on the same display. Every group of pixels represents a range of values that covers several data objects at the same time. The aggregation of the data values defines the final pixel color [MK08]. It should be noted, however, that more than one pixel are used for the representation in practice to make the area large enough for the user to interact. In contrast to many other techniques the interesting information is hereby contained in frequencies. Pixels that relate to similar value sets can be, but do not need to be contiguous. Repetitions in well-defined horizontal or vertical distances also indicate correlations. The human recognition is able to detect patterns in complex structures even if the data is distorted by noise. While pattern detection is easy, interpreting their meaning is often challenging.

2.4.3.3. Projection Methods

The group of projection methods reduces the data space in order to represent it in the 2D visual space. The data space describes the set of all possible combinations of different data set attributes. The projection tries to map the information that is inherent in this high-dimensional space into 2D. Typically, this is performed on the base of some notion of distance, which they aim to preserve in the screen space. Scatterplots are visual projection methods that are rather easy to understand. The main advantage compared to other techniques is their simplicity and the fact that most users are familiar with this concept already.

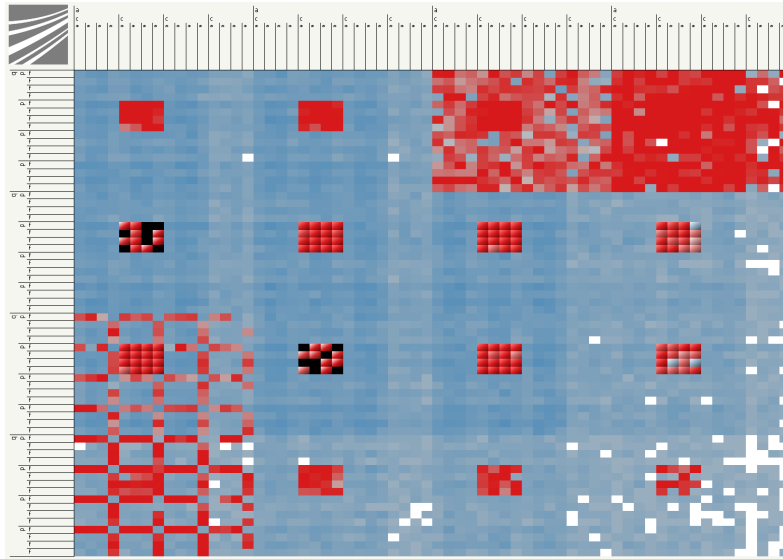


Figure 2.8: The KV-Map, a pixel-based visualization of May et al. [MK08] (with permission). A high-dimensional data set is mapped into a regular grid.

Most projection methods can be described as an optimization process that tries to minimize an objective function. The Principal Component Analysis (PCA) – a linear method – is one of the most prominent approaches. It describes the variance of points along one or more arbitrary axes in space. All of the axes are perpendicular in multi-dimensional space. In case a data set contains structures with an intrinsic dimensionality that is higher than the target dimensionality or not along straight lines, linear projections often fail to represent the data set properly. Schreck et al. present a projection method that is based on Self-Organizing Maps [SBvLK09]. This type of method is sometimes also referred as Kohonen maps, named after Teuvo Kohonen. Figure 2.9 illustrates the mapping of a 10-dimensional data set into a regular sampling grid. As the name already implies, the maps are self-organizing neuronal networks that map high-dimensional attribute space in the two-dimensional display space. In contrast to other methods, the display space is discrete rather than continuous. Every data element is represented by an element that belongs to exactly one of the classes (i.e. cells in the grid). Every class contains one element that represents the class as a whole. The classes are related to with each other in terms of similarity: classes with similar content are also close in the map.

Linear projection methods typically work with numerical data. Non-linear projection methods are able to work with other data types if the spatial distance between two data elements is defined. Above all, projections describe the data distribution in a multi-dimensional space. As a result, the points are mapped so that elements that are close in the data space are also close in the 2D space. Thus, these methods are particularly useful for clustering, similarity detection and outlier detection. We will go into more detail on projection methods and their applicability for similarity detection in Section 4.6.

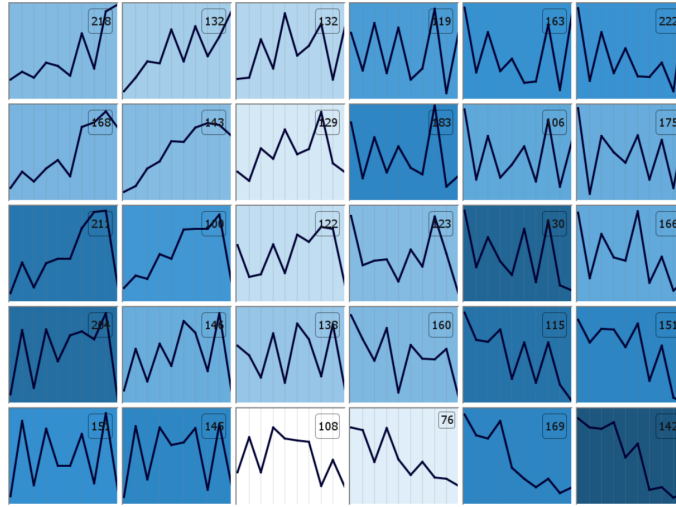


Figure 2.9: A Self-Organizing Map that depicts a 10-dimensional data set. Each of the cells represents a group of data items, illustrated by a Parallel Coordinate Plot. The number of contained elements is printed in the top-right corner of each cells. The background color indicates the quantization error, i.e. how well the plotted PCP represents the elements in the cell. Image taken from Bernard with permission.

2.4.4. Hierarchies and Trees

Network data and visualizations are also part of Shneiderman’s taxonomy. They are highly relevant for this work and are therefore discussed in detail in Chapter 3. Trees are a specific subgroup of such networks that describe binary relations, often in an hierarchical manner. This makes them different from general networks and this is why they are typically depicted by very different techniques. Their visualizations exploit their simple structure, especially the fact they typically describe orderings. For trees (graphs in which any two vertices are connected by exactly one path) the classic layouts will position children nodes below their common ancestor [RT81]. In 3D diagrams, a cone layout can be used [RMC91].

Most approaches in terms of visualization expose the hierarchy as dominant structure although several other attributes of the elements are present in the visualization. As the hierarchy does not impose a particular spatial structure, visualization techniques can be separated in two distinct parts. The first group deals with the design of visual mappings, i.e. the selection of attributes and metaphors for the display of elements and their connections. The element position in the 2D space does not play a major role for them. The second group is dedicated to different layout algorithms that map the elements according to one or more properties into the visual space. The working group of Keim presents two space-filling methods that display hierarchies in different manners [HDKS05, MKN*07]. The first one displays child nodes in their own separate space whereas the latter uses – similar to a treemap – the space of the parent node.

Among others, the importance of leaves compared to inner nodes has influence on which one of the two methods makes more sense. The treemap puts the focus on the leaves of the tree. In

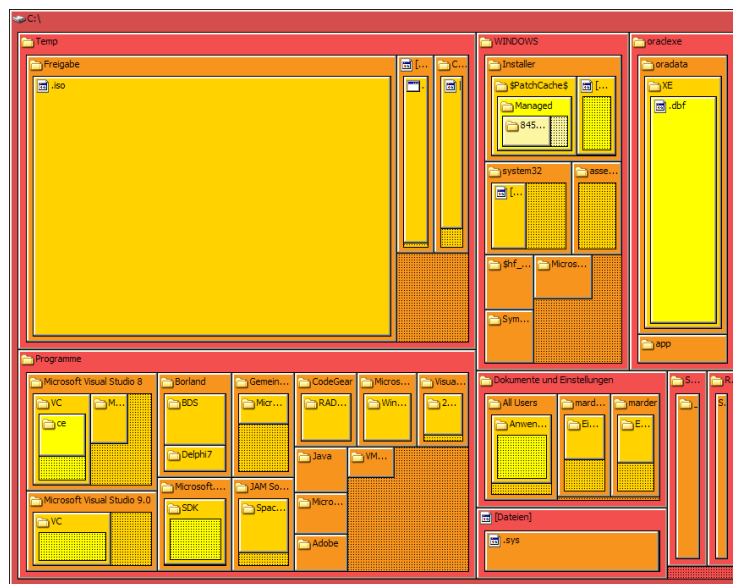


Figure 2.10: A TreeMap visualization that shows the contents of a computer hard disk as a nested structure. The size of an element on the screen relates to the size on the disk. The image is in the public domain. Source: http://en.wikipedia.org/wiki/File:Tree_Map.png.

contrast, the hierarchical layout highlights nodes that are close to the root node and less dominant in the treemap. Holten gives an example of a combination [Hol06] with a graph visualization. A node-link diagram is shown on top of a hierarchy with different aspects of the data. The edges between nodes are gathered in bundles in order to reduce the overdrawing and thus increase the readability of the graph. A simple variation of a tree layout diagrams is the traditional Dendrogram. It is characterized by the fact that all nodes of a hierarchy level are in the same line. This significantly improves the visual arrangement of the tree. The simplicity of the structure and the display allows more complex presentation of information. See Figure 2.11 for an illustration.

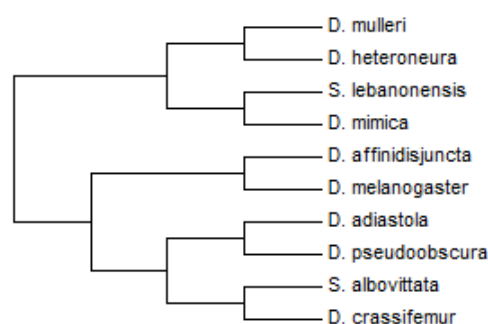


Figure 2.11: A dendrogram showing the classification of *Drosophila*, which are commonly known as fruit flies (own drawing).

Up to a certain point it is possible to create abstractions of the components and use more or less independent techniques to display nodes, edges and the structure itself. The arising number of combinations is thus a source of new designs even without fundamental novelties. Facing aesthetic, scientific and task-related aspects, designs tend to become overly complex which is conflicting with the user's need for visual interfaces that are easy to understand and learn. A meaningful visualization provides the relevant overview information on first sight without need for the user to actively search for it. This conflict has been actively discussed in the scientific community in the past years [Lor04, vW05]. The task defines the data that should be displayed, but it inherently defines the amount or type of data that should be hidden from the user as well. The data types impose a natural limitation on the repertoire of visual mappings.

2.5. Visual Interaction

Hearst gives an overview over the most important techniques for interaction and navigation in the information visualization domain [Hea99]. The combination of visual metaphors and interaction methods build the foundation of *interactive* visualization techniques. They can be classified using the Card-Model (see Section 2.3) with respect to the step in the visualization pipeline they act on. We will briefly discuss the classification of Hearst, but also add *Direct Manipulation*, because it is often used as a basis for other interaction techniques.

2.5.1. Direct Manipulation

Strictly speaking, direct manipulation is not an interaction technique in the sense that it manipulates a certain step in the visualization pipeline. As the name inclines, it defines interaction based on the screen coordinates.

Indirect manipulation can be seen as one possible type of direct manipulation with respect to GUI elements, because it immediately affects them. On the other hand, it is indirect with regards to the actual visualization, which is configured through the GUI elements. A clear definition of the parameters enables an equally clear separation of the inherent functionality and its controlling mechanisms. However, there is always a trade-off between the complexity of the functionality and the general usability. The means of this manipulation does not necessarily correspond to the effect they cause. For example, moving a slider along the horizontal axis does not directly relate to the affected parameter setting underneath. Shneiderman presents techniques to bridge this mental aiming for intuitive user interfaces [SPCJ09].

2.5.2. Brushing and Linking

The idea of *Brushing and Linking* is to generate a visual connection between two or more views on the same data entity. Selecting or hovering an entity in one view also triggers a visual change in the other, linked views. This obviously requires processing the data elements in multiple visualization pipelines, resulting in different visual representations. It also requires an inverse mapping of the transformation: the system must be able to identify the element of interest based on screen

coordinates. Therefore, visualization techniques that support Brushing and Linking offer methods to assign pixel coordinates to one or more data elements. The actual functionality is mostly independent from individual visualizations, because it uses only references to the data elements. On the other hand, the visualization system as a whole needs to provide support for exchanging such information between different views. Brushing and Linking increases the benefit of individual visualizations by coupling them, thus allowing for analysis from different perspectives. One prominent example is the *Snap-Together Visualization* of North and Shneiderman [NS00].

2.5.3. Panning and Zooming

Panning and Zooming is a group of techniques for the manipulation of the view transformation of visual structures. Most of these interaction techniques use the metaphor of a virtual camera to navigate in two- or three-dimensional space. Changing the position of this camera with respect to the screen canvas is referred to as *panning*. Zooming refers to moving the camera towards and away from the canvas. This often comes with an adjustment of the number of visible details. Bederson and Hollan demonstrate this approach with their Pad++ tool [BH94]. Today, panning and zooming is particularly often used in graph visualization techniques. In abstract terms, zooming can be seen as selection of a subset of the domain which requires the visualization to display this information accordingly. In environments with more than three dimensions, the camera metaphor is not always ideal. It is therefore often combined with dimensionality reduction techniques. While the general idea of panning and zooming works in principle for any visualization, there are other methods that adjust the perspective on the data better. One possibility is to adjust the zoom factor for the axes individually, for example in parallel coordinate techniques.

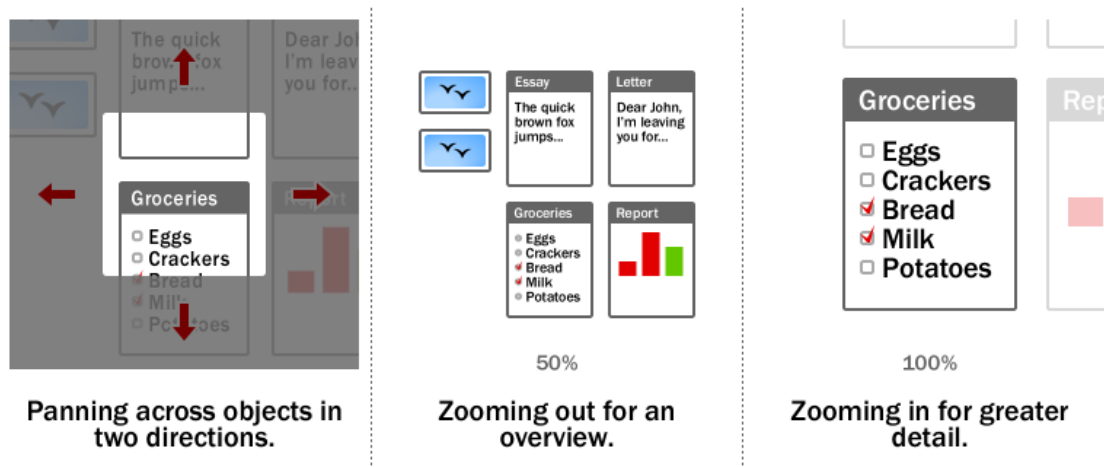


Figure 2.12: An example of a zoomable user interface for illustrative purposes. The user can pan across the interface, zoom in to get more detail and zoom out to gain a better overview. Some user interfaces adjust the level of presented detail to the zoom factor. The image is in the public domain. Source: http://en.wikipedia.org/wiki/File:ZUI_example.png.

2.5.4. Animation

Fisher describes the principle of animation as “[...] a series of images trying to build a coherent idea of what occurred between them” [Fis10]. In a user study, he compares animation with small multiples (a collection of “key frames” put in juxtaposition) for different task. As it is often the case, the quality of the results depends on the user groups and the task. Fisher states that animation tends to take more time and is less accurate in explorative tasks, but outperforms small multiples in presentations.

Animation can be considered exclusively as part of the visualization, but also as part of the interaction concept. Due to its nature, it is often used to display time-dependent data. In this application, animation can be exploited to spot significant transition patterns over time [TK07]. In this case, no interaction is involved. Interactive visualizations on the other hand use animation to communicate changes based on user interaction. To give an example, the *DynaVis* framework of Heer and Robertson animates changes between different statistical visualizations [HR07]. Formally speaking, the parameter space is interpolated between the value before and the value after the user interaction. Animation supports the user’s orientation in situations where changes in the configuration of the visualization affect the visualization as such. It preserves the perception in transitions between two consecutive displays of the same data element. The human eye is able to track the transformation. The cognitive load on the user is reduced by providing a consistent visualization of data items and exploiting the human perceptual system [RM93]. We will make use of this fact in the visualization technique that is presented in section 3.6.

2.5.5. Focus plus Context

The group of Focus plus Context approaches is strongly related to one particular problem of zooming. The larger the zoom factor is, the more details can be displayed on some few entities of interest. The downside of this approach is that the overall structure and the location in this larger context are no longer visible. Focus plus Context aims to overcome this limitation by separating the displayed information into a detailed focus and a less-detailed context area. This is similar to the concept of Overview plus Detail (see Section 2.5.6), but works with one single view. Prominent examples in this category are distortion-based visualizations such as the FishEye view [Fur86] and TableLens [RC94]. Kosara et al. present an entirely different method which uses selective display of information [KMH01]. With the help of virtual depth-of-field blurring, relevant information is highlighted without compromising the ability to show an overview of the situation. Many Focus plus Context techniques work on the basis of an arbitrarily shaped area of the screen – the *lens* – as a metaphor for interaction on data elements within this area. Typical applications are optical zoom, but also the display of additional or otherwise hidden information [Hea99].

Adjusting the level of information according to a (virtual) zoom factor is subject of *Semantic Zooming*. In contrast to optical zoom, the structure of the displayed data is modified rather than the parameters of the graphical presentation. The number and the granularity of visual elements are adjusted with respect to the desired level of detail. This includes additional graphical metaphors such as annotations and glyphs that are inserted into the display. From an implementa-

tion perspective, this requires that a mapping exists that defines the visual representation for every object type for every level of detail. In the survey of Cockburn [CKB08], several aspects of the user's focus in graphical interfaces are discussed. One of the conclusions of this work was that Semantic Zooming can reduce the task completion time in map-based navigation tasks.

2.5.6. Overview plus Detail

In contrast to Focus plus Context techniques, *two or more* linked visualizations with different zoom factors are used in *Overview plus Detail* approaches. Card et al. note that the ratio of the screen size of two displays range is mostly between five and fifteen [CMS99a]. The authors differentiate between temporal and spatial techniques, depending on whether the visualization displays overview and details one at a time or at the same time in different views. Cockburn et al. define such interaction interfaces as “..the simultaneous display of both an overview and detailed view of an information space, each in a distinct presentation space.” [CKB08].

Temporal Overview plus Detail views are conceptually similar to *Semantic Zooming*. Spatial Overview plus Detail on the other hand uses two or more separated displays rather than different zooming levels in the same display. The technique helps users to keep an overview on the entire structure while looking at a small portion of the data at a more detailed level. Interestingly, this metaphor is often used in map-based games in the form of a *minimap* that puts the camera's view on the map in relation to a larger (annotated) context. An early example for Overview plus Detail visualization systems is the *Harmony VRWeb 3D* scene viewer [And95]. It uses a 2D-map for browsing through search results in an information landscape.

2.6. Visual Analytics

As has been illustrated in the introductory chapters, the amount of raw data increases exponentially. This comes along with the increasing computing power. Human capabilities on the contrary remain constant. As a consequence, the assessment of data becomes more and more time-consuming. Even more serious is the fact that it becomes increasingly difficult to make sense of the data, find overarching patterns and links. This group of problems is often referred to as *information overload*.

Different solutions have been proposed to keep up with the increase of raw data. In particular, information visualization has been employed successfully to enhance human capabilities of sense making and decision making based. A key factor is the support of internal models of the data through external representations. Fekete et al. state that:

“Visuals augment human memory to provide a larger working set for thinking and analysis and thus become external cognition aids” [FvWSN08].

The aim of interactive data analysis is to extract or generate knowledge from data sets with the help of the user. This can be done by manual parameter tweaking, text-based interaction or visual interfaces. *Visual Analytics* is about combining the advantages of human visual recognition with the strengths of machine-based computation. However, human's visual intelligence can be used

only if adequate data displays are provided. The contribution of this dissertation is settled in this research field which is highly focused on interactive visual analysis.

2.6.1. Related Research Fields

In this section, we will describe the term and set it in relation to adjacent fields. Basically, Visual Analytics has evolved from the domains *Information Visualization* and *Scientific Visualization* [KMS*08]. One of the first publications on the subject was the presentation of “Waldo” by Keim et al. [KPSN04] in 2004. It is a software tool that combines data mining with interactive visualization to analyze large (geospatial) data sets. A major driver for Visual Analytics was the book “Illuminating the path” by Thomas and Cook that appeared one year later [TC05]. The authors state that:

“Visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces” [TC05].

This is related to the definition of Visual Data Mining (VDM). Simoff et al. state that “Visual Data Mining enables discovering of regularities in data sets visually or regularities in the output of data mining algorithms [...]” [SBM08]. The human perception system is used to identify visual patterns that are created by data mining algorithms. Even today, this task still poses a major challenge for computer algorithms. Consequently, the human can be seen as a part of the data mining pipeline. In Visual Analytics on the other side, the human user is also recognized as steering entities that should be equipped with powerful interaction tools for the analysis task.

Both analysis pipelines have the same goal: gain knowledge and insight from a potentially large data set or confirm an existing hypothesis. In contrast to the conventional data-mining, the strong involvement of the user plays a major role in both processes. Consequently, the goal is reached in an iterative process that is guided by the user that steers of the computation. This is also highly correlated with the idea of *Computational Steering* [WBD00]. In our opinion, the latter is more focused on optimizing simulation processes through human-computer-interaction (HCI). While Visual Analytics is the overarching theme for this work, we will discuss Computational Steering in more detail as part of the network analysis tool that is presented in Section 4.8.

In the work of Keim [KMS*08], the field of Visual Analytics is revealed as a highly interdisciplinary one. It overlaps with many other research areas such as Knowledge Discovery, Geospatial Analytics, Cognitive Science and Data Management. Figure 2.13 illustrates Visual Analytics in the context of other fields as seen by Keim.

2.6.2. The Visual Analytics Process

The main idea of Visual Analytics is the combination of machine-based computing with human recognition and sense making. Data is processed by both visualization tools and automated methods. The two components are linked by user interaction to enable tuning the model generation process. We will briefly describe the iterative Visual Analytics process that is depicted in Figure 2.14.

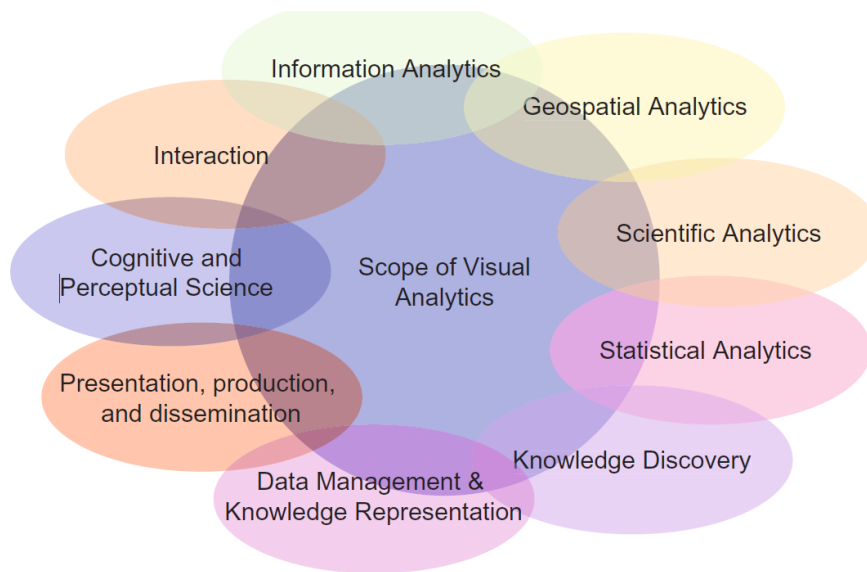


Figure 2.13: Visual Analytics and related research fields as defined by Keim et al. [KMS*08]. The image is in the public domain. Source: <http://de.wikipedia.org/wiki/Datei:VisualAnalyticsOverview.png>.

The input data is not restricted in any way. It can come from different sources (the *input*), it can be structured (tabular) or unstructured (e.g. text collections). The data is then transformed using different, pre-processing steps that (iteratively) clean, select or aggregate different attributes. This data can be used in two ways: for (automated) model generation and for visualization. Note that the model generation can also be performed based on visualization. As soon as the model is created, it can be refined iteratively, for example through parameter adjustments. The model can also influence the visualization. This means that both data and the model was derived from the data can be used in the visualization. Also the user plays a role here by interacting with the visualization. This can lead to further model refinement and closes the loop between visualization and model.

The output of the process is *insight* – often also referred to as *knowledge*. It can be reached either through the visual access or automated algorithms. Through the *Feedback Loop*, gained insight is put back into the system and process can restart in a refined manner. As can be seen in the diagram, the Visual Analytics process is highly iterative at both a high level structure, but also with respect to individual steps. The loop between visualization and model building plays a key role.

In [KMS*08], the expression *hypothesis* is used instead of *model* as in [KAF*08]. We prefer the model-based definition, because this better supports machine-based approaches. Knowledge discovery is about model building. In contrast, only human can create a hypothesis. Also, models can be parametrized which is not the case for hypotheses. Keim et al. define the Visual Analytics mantra as follows:

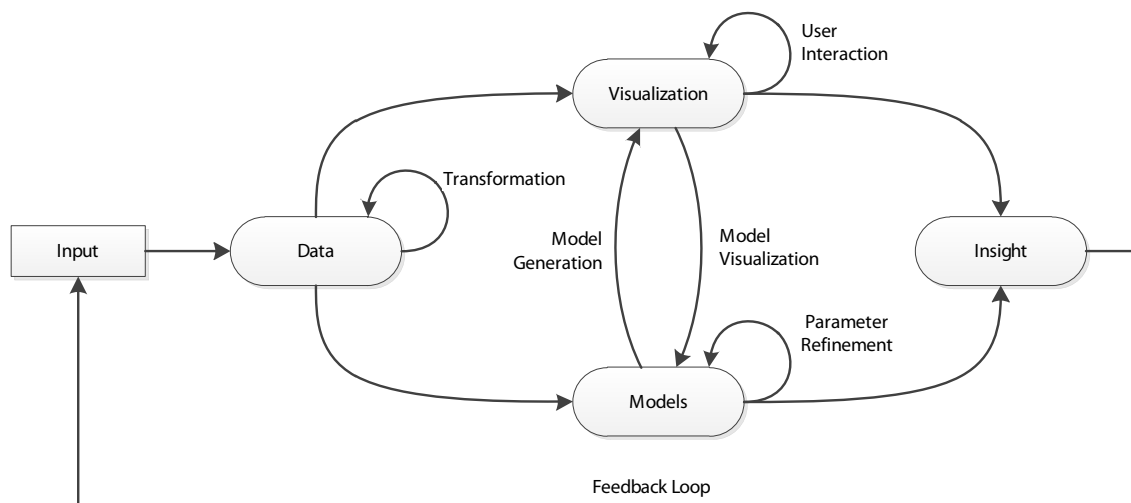


Figure 2.14: The Visual Analytics Process as defined by Keim et al. [KMS*08, KAF*08] (own drawing)

“Analyze first – Show the Important – Zoom, Filter and Analyze Further – Details on Demand” [KMS*08]

This is in line with Shneiderman’s information seeking mantra, starting with an overview of the data set based on the most important aspects. Then, the user is integrated in the process in the drill-down phase where filters and focus are defined. This allows for a more detailed analysis of specific parts. In a final step, the user can access details on demand. The process is not a one-way pipeline though. The user is free to navigate back and forth as desired to refine the search and explore the data as needed. We will build upon this model in the next chapters of this thesis, in particular on the idea of integrating the user tightly in the process of information extraction.

2.7. Summary

This section covers the foundations of Information Visualization. Starting with a retrospective on the most prominent works and drawings, the tasks visual analysis tool users typically need to accomplish are discussed based on a multi-level task typology. The contribution of this thesis is most strongly linked to the *exploration task* that is located in the *Search* category in the model of Brehmer and Munzner [BM13]. Then, the data flow pipeline of Card et al. [CMS99b] is presented. It can be seen as the overarching architecture that the visualization systems that are presented in the following chapters follow.

Shneiderman came up with a classification of different data types [Shn96]. We use it in combination with Keim’s taxonomy [Kei00] to identify the visualization and interaction techniques that fit best for the different use cases. The category of graph data is discussed in more detail in the next chapter. Linking back to the introductory chapter, we describe the powerfulness of the Visual

Analytics approach that we use to extract valuable information from data and graphs in particular. All of the contributions in this thesis live under this premise.

3. Local Graph Views

The focus of this chapter is on *Local Graph Views*, a collection of visualization techniques for the exploration of graph structures. It contains the first group of contributions of this thesis and also sets the foundation for the second group of contributions, which are discussed in Chapter 4. Starting with a Focus plus Context technique that provides visual cues for orientation and navigation, several contributions are outlined that aim at preserving the user’s mental map in the exploration process. This enables domain experts to work with large networks, find relevant connections between different parts of the graph and gain detailed insight in the data.

In contrast to other graph layout approaches, the idea of Local Graph Views is to focus on a small subset of the data instead of trying to display the entire structure as a whole. This makes it particularly useful for large data set as the approaches typically scale very well. One example is depicted in Figure 3.1, where the user explores a medical database. Only a few nodes are visible. Selecting interesting nodes marks them as focus nodes and adjacent nodes appear in the display. Visual cues point towards interesting part and thus support the user’s orientation and navigation. This enables the analyst to browse through the network.

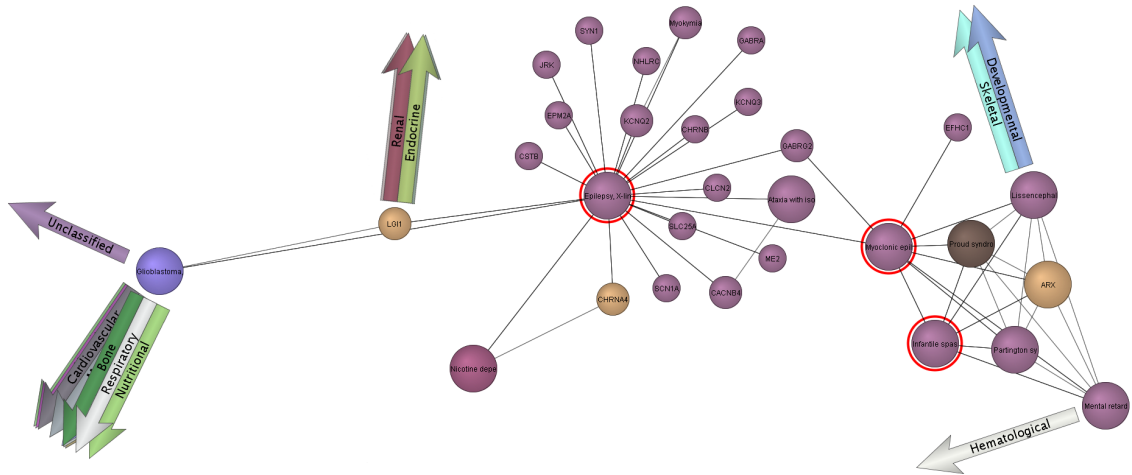


Figure 3.1: Navigation in a large graph from the medical domain [GCV*07] using signposts to provide context for the nodes in focus (as used in our user study). The focus is derived from the focal nodes (encircled in red).

We pick up many of the ideas that are presented in Chapter 2, in particular the following concepts: Card’s data flow pipeline is used as a baseline throughout all the presented approaches. Also, projection methods play a major role in several visualization techniques as they map com-

plex high-dimensional objects into the two-dimensional screen space. A variety of the mentioned interaction techniques are used in different scenarios. Most importantly, the Focus plus Context concept plays a key role for local graph views.

The chapter commences with an overview on the problem domain and sheds light on the most relevant problems from a practical perspective. On that basis, a set of five requirements for the successful analysis of graph data in local graph views is defined in the following section. Then, related work is discussed, starting from a high-level perspective and narrowing down on the most similar approaches. Before we go into details on the individual contributions of this part of the thesis, some necessary definitions are presented in a brief overview.

The first contribution is about adding visual cues to the visible part of the network to support the user in finding interesting parts of the network and help with orientation. A case study indicates that these cues are helpful. In a follow-up, a group of techniques to support the preservation of the user's map in the exploration process by reducing the node movement. Experimental results confirm that this is achieved. This is strengthened further with two approaches that make the exploration process deterministic. Again, this was verified with experiments on real-world data sets. In the last section of this chapter, we look back on the set of requirements to see if they can be fulfilled with the presented contributions.

Parts of this chapter have been published in [MSDK12,SMK13,SMK14,SLTM*13,SLTM*14].

3.1. Background

The tasks in the 2012 VAST Challenge¹ are about finding suspicious events that happened during just two days. The given data set describes states and connections of all computers in a fictive company network. The major problem here as well as in most real world scenarios is the sheer amount of data. In total, 160 million nodes are available in the database. A visual representation of the data is vital to gain insight and decide what is suspicious and what is not. Obviously, displaying all entities and all interconnections at the same time cannot produce meaningful results. Screen space is always limited, especially compared to the ever-increasing amount of graph data, so massive overdraw is the consequence for such visualizations. But also visual perception has its limits. Even if it would be possible to draw millions of nodes, the data analyst would not be able to deal with that information. Typically, the user is interested in a small part of the graph, maybe even a single node. The challenge in the visualization is to preserve a context that is large enough for the user to orientate and navigate in the abstract spaces of graphs.

The display of networks helps to analyze relationships between entities rather than the entities themselves. Graph visualization has become an important topic in information visualization area over the past years and is applied in many different application areas today. For example, the site maps of web sites as well as the browsing history of a web browser can be displayed in a directed graph. In biology and chemistry, graphs are applied to evolutionary trees, molecule structures, chemical reactions or biochemical pathways. In computing, data flow diagrams, subroutine-call graphs, entity relationship diagrams (e.g., UML and database structures) and semantic networks and knowledge-representation diagrams are the main application fields. Furthermore, document

¹<http://www.vacommunity.org/VAST+Challenge+2012>

management systems profit from document structure and relationship visualization. Social networks visualization has also become a popular application of graph visualization methods.

Who are the users of such software tools? We differentiate between casual and expert users and focus on the latter groups that will be referred to as *data scientists* from now on. It is important to differentiate between users that are experts with respect to computer science and those who are experts in a particular application domain. Users are only rarely familiar with both domains. For real-life applications, users know their application domain quite well, but are not computer-savvy. The focus of this work is on such real-world scenarios and we therefore target a user group with only little computer experience. As a consequence, visual tools must be intuitive and easy to follow.

The tasks we support with our technique are characterized by the following assumption: Information on the local level is more important than the global structure of the graph.

Showing the structure emerging from the network connections is one goal of graph visualization. However, human's visual intelligence can be used only if adequate data displays are provided. Node-link diagrams are a popular visualization techniques that works particularly well for small to medium-sized graphs. The goal of our approach is to ease the exploration of local structures of large static graphs based on such a node-link diagram.

Graph analysis has gained strong interest due to the fact that in contrast to tabular data, graphs also contain explicit links between data items. One type of visualizations for graphs is the node-link diagram with its manifold variations. Despite their conceptual simplicity, they produce pleasing drawings for small to medium-sized graphs. For large graphs, however, they do no longer scale and the drawings become tangled up.

One effective counter-measure is to use *dynamic views*, in particular for explorative tasks. One or more filtering criteria are applied to such a large graph and to extract a small subgraph which contains the most relevant or interesting items. Here, the focus of the data analyst is on local features rather than on the global structure of the graph. Instead of trying to create a complete picture of the entire structure, dynamic views work with a selected sub-graph of the original data set. This implies that local structures are preserved at the expense of global structures. This filtered view is then used for the analysis instead of the large one.

3.2. Requirement Definition

We define a set of requirements that must be fulfilled to support the data analyst to successfully solve tasks based on graph data sets.

For small to medium-sized graph a pre-computed global layout with fixed node positions has its advantages. Above all, the visualization of a specified graph will always look the same which helps to build a mental map of the data set. In the navigation process, the nodes are simply toggled between visible and invisible state while their positions remain the same. But this also means that nodes that do not exist in the partial view influence the position of the visible nodes. This effect can be seen in Figure 3.1: some nodes are strongly pulled outwards without any visible explanation. They have to fulfill all constraints that are imposed by the full graph.

The time and memory limitations render global layout impractical for large graphs. Also, a fair amount of the computed layout is never used in the navigation process later on. One way to approach this problem is to perform the layout computation on-the-fly for the area that is currently visible. This process works quite well, if the displayed subgraph does not change over time. Otherwise, these topological changes can have a strong and sudden impact on the visual representation if they are not carefully dealt with. In order to communicate the performed changes it is important to create smooth transitions, for example through animation, so that the user can better understand which nodes appear or disappear.

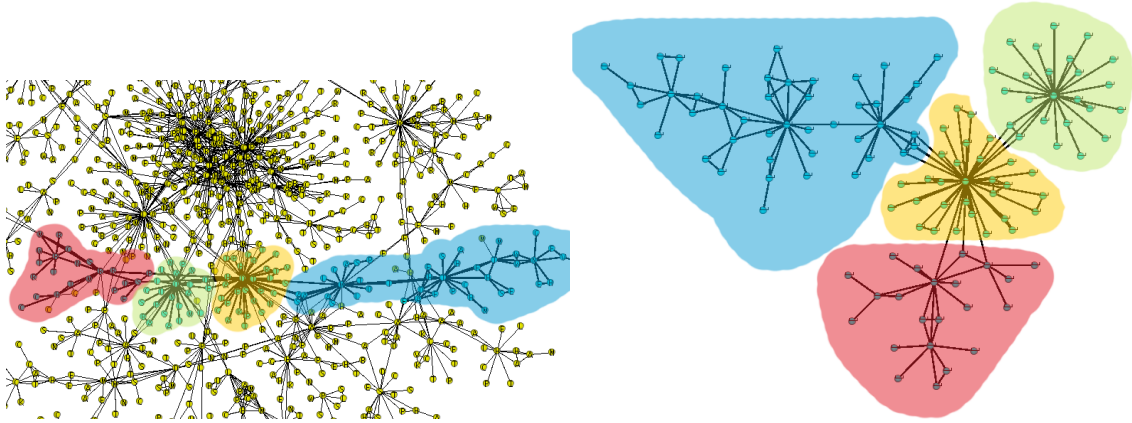


Figure 3.2: *Identical network parts have been highlighted in a global layout (left) and a local, independent layout (right). The former display makes the impression that the clusters were linearly connected. In particular, it seems as if the shortest path from red to blue leaded through green and yellow. However, the local view reveals that this is not the case.*

The first requirement is to show only the part of the graph the user is interested in and to adapt the layout to this visible subgraph. To give an example, Figure 3.2 shows a global layout with four clusters in a linear arrangement. Instead, a local layout of the clusters exposes the actual topology of these clusters.

Requirement 1 (Complexity Reduction) *Reduce the amount of visual complexity to a minimum.*

Today’s data scientists need to work with different types of graph data. Apart from company networks, many applications ranging from chemistry, physics, life sciences and many others heavily rely on the analysis of relations in these data sets. Prominent examples are investigations in citation networks or social networks.

Let’s look at an example that is settled in the medical domain. We know today, that many diseases are caused by genetic defects. Looking at it from the opposite side, certain genes are responsible for many congenital disorders. The analyst investigates these diseases with respect to their topological neighbors. From this, we derive our second requirement as follows: the user needs to know in which part of the graph the current focus is and how it relates other parts of the network.

Requirement 2 (Orientation) *Preserve orientation in local graph views.*

Some of those diseases are linked by common genetic properties. Many of them are related to the same body region or belong to the same group of disease, but interestingly enough, some of them connect different groups. These connections can be of particular interest for data analysts. The user wants to find out, which connections exist between certain regions or groups to identify common causes. By default, these highly interesting bridges do not look different from the rest. This makes navigating towards potentially interesting nodes hardly feasible. Consequently, the user needs visual support in finding the right paths to be able to navigate between regions.

Requirement 3 (Navigation) *Support the user in navigating between different regions of the graph.*

Dynamic layout also included the smooth transition between consecutive layouts of these visible subgraphs during exploration. Already in 1991, Eades et al. note that the visual order of screen elements should not change for animated diagrams in order to prevent user irritation [ELMS91]. We agree with the authors: the exploration process must provide smooth transitions between different views so that the user can keep track on changes in the node-link diagram. The rationale for smooth transition is to minimize the users' effort to keep track of the evolving layout.

Requirement 4 (Smooth transitions) *Support the user in the navigation process in an unobtrusive way, avoiding irritations and distractions from the actual task.*

A typical user task is the exploration of the neighborhood around a focal area of the graph. We consider such an exploration as success if the user is able to mentally chart the visible parts of the graph. Thus, it increases the area the user is familiar with. To be precise, "familiarity" reflects the ability to recall a visible area upon revisiting and to mentally extend this area beyond the visible part. This enables the user to plan and predict navigation. Lee et al. [LPP*06] identified *Revisitation* as one of the tasks to be supported by graph visualization. However, we believe that this task is not yet supported enough in current research.

The "smooth transition" approach applies to consecutive frames only. Revisiting a known area of the graph after extensive exploration usually results in very different layouts. This fact leads to our next requirement.

Whenever a user revisits an area of the graph for a second time the difference between the two layouts should be as small as possible. We state that this requirement applies regardless of the length or direction of the user's exploration path between any two visits of the same area. For static graphs we argue that the visible layout is determined by the currently visible subgraph only. A simple solution exists if only the second requirement were to be considered: Compute a *static* layout of the complete graph and toggle the visibility of nodes and edges as needed. However, a static layout conflicts with the first requirement (1), because it naturally does not adapt to local features.

Requirement 5 (Determinism/Familiarity) *The user is able to recognize a previously visited area of the graph.*

The overarching goal is to preserve the mental map during the exploration process. These requirements must be fulfilled to enable successful exploration of graph networks. We propose solutions to all of these requirements.

The first one (Preserve local features) is subject in all of our approaches and in a number of existing *dynamic layout* approaches. These cover techniques for the selection of interesting or important areas of the graph, for the definition of incremental layouts. Requirements 2 and 3 are discussed in Section 3.5, Requirement 4 in Section 3.6. We resolve the conflict of the requirements 1 and 5 (the preservation of local features versus the reproducibility of layouts) explicitly in Section 3.7 and improve the stability of the approach in Section 3.6.

3.3. Related Work

In this chapter we discuss scientific publications that are related to this work in one way or another, grouped by topic or category. Our approach is related to previous approaches from different areas including navigation concepts in visualizations, off-screen visualizations and multi-level approaches to graph drawing. We will go through each of the areas and present the similarities and differences between existing approaches and ours.

The key issues in graph visualization are the graph structure (directed vs. undirected graphs, trees vs. cyclic graphs) and their size. A survey of graph visualization techniques for different graph types can be found, for example, in the work of Herman [HMM00]. Von Landesberger compiled a state-of-the-art report on visual graph analysis specifically for large graphs [vLKS*11]. It covers a variety of analysis techniques for node-link diagrams as well as matrix representations for partial graph views or dynamic graphs.

In Section 3.3, a graph was defined as a set of nodes and a set of edges (i.e. node pairs) that connects some of these nodes. Two, very complementary approaches exist to display such data sets: *node-link diagrams* and *matrix displays*. Matrix displays display edges as entries in a grid. The ordering of rows and columns of these matrices plays a major role for effective identification of clusters and many other tasks.

In node-line diagrams on the other hand, the node layout defines the visual representation. Relationships between nodes are typically drawn with line segments. There are different graph layout techniques suited for different graph types.

3.3.1. Adjacency Matrix Displays

Displaying the list of connected nodes in a two-dimensional grid is a straightforward approach. If n being the number of nodes, the size of this matrix is $n \times n$. This also means that the size is independent from the number of edges and therefore well suited for dense networks such as small-world graphs. The entry in row i at column j indicates if node i is connected to node j . Entries on the diagonal are typically left out, because reflexive edges (i.e. self-loops) are prohibited by definition. We note that this matrix is symmetric for undirected graphs. Figure 3.3 shows an example.

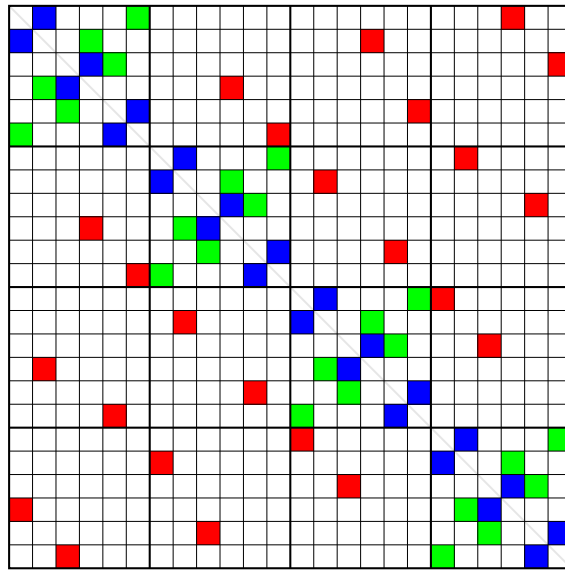


Figure 3.3: An example of a graphic representation of a symmetric adjacency matrix (the Cayley graph of S_4). Colored fields define links between nodes. This matrix is symmetric, which is the case for all undirected graphs. The image is in the public domain. Source: http://commons.wikimedia.org/wiki/File:Symmetric_group_4;_Cayley_graph_1,5,21_%28adjacency_matrix%29.svg

This tabular display can be used to identify individual connections between nodes, but also higher level structures can be identified. For example, a filled row i indicates that the node i is a star node. A filled triangle that is aligned at the row and column axis represents a clique (i.e. a fully connected cluster). In order to be able to identify such structures, the matrix must be sorted with respect to certain criteria. This sorting process poses a major challenge as the sorting criteria for different tasks are often conflicting and thus subject of current research. According to Ghoniem et al., matrix visualizations are superior to node-link diagrams for many tasks and for large graphs. However, a major shortcoming is that paths of connected nodes can be difficult to identify and the analysis of neighborhoods is more difficult [GFC05].

3.3.2. Node-link Diagrams

In contrast to matrix displays, node-link diagrams are very intuitive. Nodes are drawn as individual elements on the canvas, often as circles and the connecting edges as (curved) line segments. An example can be seen in Figure 3.1. In many cases, star nodes, clusters, isolated sub-graphs and other structures can be identified with only little effort. However, the choice and parametrization of the layout algorithm is vital for the quality of the diagram. Also, the number of nodes and edges is somewhat limited: with increasing graph density the display becomes more and more crowded.

3.3.3. Force-directed Layouts

Classical force-directed approaches have been researched but also used in practice for more than two decades now. Many combinations of techniques for the graph structure and the detail view are possible. Displaying details in the graph makes sense only if the information can be classified and processed on first sight, for example a mapping on a color scale. The number of currently available visualization indicates already that there is no single best visualization, neither for the graph layout nor for displaying nodes and edges. For large graphs, the large amount of node and link overplotting requires new visualization and clustering techniques such as (3D) hyperbolic space layouts [LRP95, Mun98]. For a formal definition of graph/networks we refer to chapter 3.4.

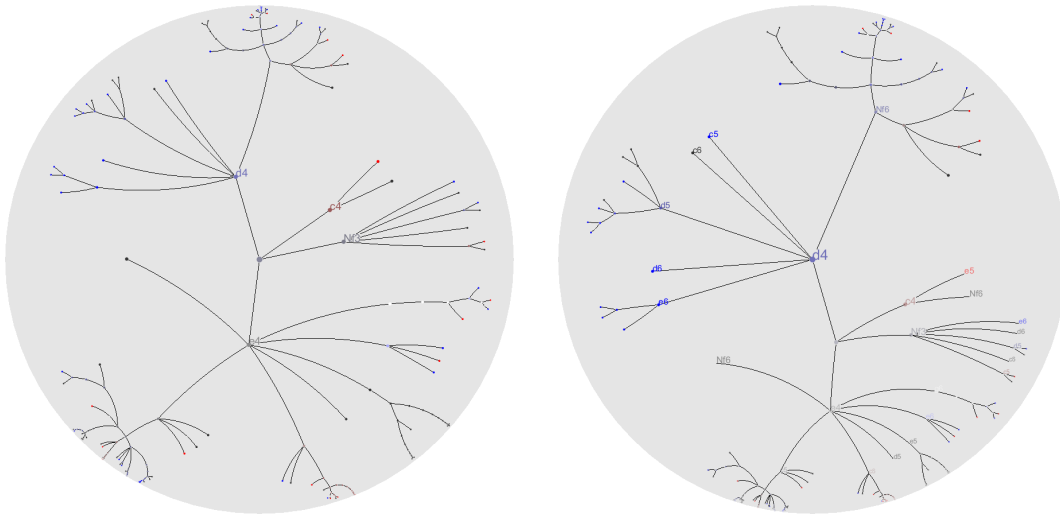


Figure 3.4: A Hyperbolic Tree (a Focus plus Context technique) reserves more space to nodes in the center at the cost of space at the boundary. Changing the focus node adjusts the amount of reserved space (right image). The images are in the public domain. Source: <http://en.wikipedia.org/wiki/File:BasicTree.png> and <http://en.wikipedia.org/wiki/File:BasicTreeFocused.png>.

The placement of nodes in an arbitrary graph layout that fulfills certain optimality constraints is quite complex, or mathematically spoken: NP-hard [BGW03]. Many graph visualizations are variations of node-link-diagrams. The publications in the domain can be split (similar to trees and hierarchies) in two categories: the graph layout on the one side and the visualization of nodes and edges on the other side. The quality of a layout is measured in different criteria which often impose conflicting constraints. It is, for example, desirable to be able to see the most significant structures and clusters. But it is also desirable to minimize the spatial distance of related partitions. This makes it per-se difficult to find a layout that is optimal for all demands. Technically, the layout is often computed by mass-spring-simulations, sometimes called “spring-embedders”. They model the optimality criteria as an energy function. The simulation then tries to find a global minimum for that function.

The first spring-based graph layout was presented by Eades in 1984 [Ead84]. Forces are computed based on Hooke’s law to ensure a certain distance between connected nodes. It also models repulsive forces analogous to the electric force to avoid node overlaps. The version of Fruchterman and Reingold improves this model by adding a “global temperature” that controls the node movement speeds [FR91]. This also ensures that the simulation terminates. Similarly, the layout algorithm of Kamada-Kawai uses the graph-theoretical distance to model the springs [KK89].

3.3.4. Projection-based Layouts

These first approaches aim to minimize energy in a virtual environment. A major short-coming of that is that this optimization process often gets stuck in local minima.

Spectral layouts are quite similar as they also aim to minimize an energy function, but they are able to find the global minimum efficiently [Kor03]. It does so by solving an eigenvector-problem. The idea here is to put a given node in the centroid of their neighbors. The same author also describes High-dimensional embedding (HDE), a group of methods that computes a layout in high-dimensional space before projecting it into 2D space [HK02]. We will discuss this in more detail in section 3.6. In an experimental study, Brandes and Pich conclude that optimizing an explicit model of desired aesthetic properties yields better results than force-directed layouts [BP09].

3.3.5. Hierarchical Graph Layouts

One fundamental problem in graph visualization is the sheer amount of nodes many data sets contain. The number of nodes that can be displayed on the screen is rather limited. Considering that the focus of the user is either on the global structure or on a particular group of nodes it often makes sense to hide a large part of the data set.

Balzer and Deussen create a visual abstraction on the basis of existing node hierarchies [BD07]. It can be, for example, generated by hierarchical clustering algorithms. The nodes and the edges of a cluster are then combined into one single graphical element. A variation of this has been presented by Henry et al. who model this graphical element as an adjacency matrix [HFM07]. Their main contribution, however, is to provide interaction tools for the user. In contrast, Gajer et al. work with a reduced set of nodes from the original data set instead of computed clusters [GGK01]. A system that is dedicated to navigate in large graphs has been developed by Abello et al. [AvHK06]. The basis for that is again a given node hierarchy. It is used to display an overview on the graph that is used for navigation. At the same time, it acts as filter for the nodes that are displayed in a detailed view. Depending on the level of detail, sub-trees are expanded or collapsed.

The complexity of network graphs is often distributed on many structural levels. Many techniques assume that it has an inherent hierarchy. They exploit that by computing and using hierarchical structures for the display. Even if a visualization technique is able to switch between different levels in the hierarchy, it is probably not able to display all levels of the structure at the same time. This does not work, though. The user’s visual ability to focus is limited to one or two

levels. An essential task of graph visualization is thus to display one structural level as good as possible and to support user-controlled switches between different levels if necessary.

Our approach in Section 3.5 basically uses a high-resolution representation for the focus and an abstract representation for the context. Multi-level representations have been used to improve different aspects of graph visualization in order to allow structural zooming within the graph. Frishman and Tal [FT04] and Huang et al. [HEL05] present different methods to improve the layout.

Our technique in Section 3.7 relates most to cluster-preserving dynamic layouts. However, the “clusters” in our approach are subgraphs, which are laid out independently in a pre-processing step and merged together depending on the current area of interest of the user. Archambault et al. [AMA07] present a similar strategy with the static multi-level technique *Topolayout*. Topolayout creates hierarchical partition layouts with the most suited technique and merges them to minimize edge lengths and crossings. In contrast, our technique creates overlapping subgraphs which are dynamically merged along the overlapping nodes.

Hierarchical layout algorithms use a divide and conquer scheme to first create a coarse layout which are refined in later steps [HK01], thus working in a top-down fashion. Using a bottom-up approach is also possible, calculating small layouts which are recursively combined until the whole graph can be drawn [PV06]. However, both approaches rely on certain properties of the division process while our top-down approach is more flexible with respect to division levels.

Alternatively, multi-dimensional layouts like that of Gajer et al. [GGK01] make use of additional dimensions to quickly create a robust layout before projecting it into two-dimensional screen space. Our approach goes in this direction, but it is not restricted to a specific algorithm. Yuan et al. propose an approach based on crowd sourcing. Users of the system manually create layouts of subgraphs, which are then automatically combined [YCHZ12].

3.3.6. Dynamic Graphs

In the literature, dynamic layout techniques have been proposed to solve two different problems: The first problem is the layout computation of dynamic graphs, which has been formalized by North [Nor96]. The second problem is the layout of a dynamic *view* of a graph, which has been described by Huang et al. [HEW98]. A dynamic view is basically a visible subgraph, which can be “moved” interactively for browsing. This problem has been applied to static graphs, for example, by Huang et al. [HEL05] and van Ham and Perer [vHP09]. The state of the art report on dynamic graph visualizations by Beck et al. gives an overview of recent developments in the field [BBDW14].

Based on North’s original definition, Huang and Eades brought up the idea to create a dynamic view of a static graph [HE98], which can be adjusted by user. We also use this metaphor to enable the user to navigate through the graph. A major shortcoming of this is the missing context, since no direct connection between the visible part and its context – i.e. the rest of the graph exists. Some techniques use the fisheye concept to surround the current focus area with a distorted view on the neighborhood [FK96, LRP95].

Many dynamic layout techniques are modifications of static layout techniques which impose specific constraints or quality objectives on the transition between two consecutive layouts. Bran-

des and Mader [BM12] compare different measures, especially with respect to the trade-off between individual layout quality and stability between frames. They note that even slightly lowered requirements in quality often offer a significant boost in stability. Virtually all elements of a graph visualization have been covered by previous approaches to stabilize the mental model upon dynamic changes. For example, Frishman and Tal [FT08] and Erten et al. [EHK*04] propose approaches where quality objectives apply to the node movement. Frishman and Tals approach fine-tunes the inertia of nodes between consecutive frames. Erten et al. propose a natural extension to force-feedback techniques by using virtual edges across different time frames. Other approaches, like that of Dwyer et al. [DMS*08] and Frishman and Tal's [FT04] focus on the preservation of node clusters in the dynamic layout. Aside from spring-embedding layouts, dynamic layout methods have also been used in conjunction with other techniques. For example, Görg et al. [GBPD05] use Sugiyama-style layout techniques.

3.3.7. Orientation & Navigation

A recent synopsis of navigation concepts has been given by Cockburn et al. [CKB08]. They categorize *Overview & Detail*, *Zooming & Panning*, *Focus & Context* and other techniques based on their interface mechanism, and give a survey on evaluations of these methods. While Overview & Detail is considered as a spatial separation into two or more distinct visualizations, Zooming & Panning is considered a temporal separation between emphasized and suppressed subsets of the data. Focus & Context techniques avoid spatial or temporal separations by creating a seamless visual embedding of detailed information (the *focus*) within its *context*.

Aside from techniques which aim to preserve the mental map on a purely structural level, the role of interaction and navigation cues must be considered as well. In fact, Marriott et al. note in their study [MPWG12] that node labels are powerful cues for mental mapping. However, we think that layout stability supports the effective use of local navigation cues like labels, because they need to be located in the view to be useful. Moscovich et al. [MCH*09] and van Ham and Perer [vHP12] propose techniques to ease navigation across larger distances. Their common idea is to provide visual cues pointing to otherwise invisible nodes or regions of the graph.

3.3.8. Preserving the Mental Map

In this section we first describe fundamental work on the preservation of the mental map for the navigation in network visualization, especially with respect to design considerations and criteria.

The preservation of the mental map of graph visualization has become an important goal ever since its introduction by Eades et al. [ELMS91]. The original definition of the term "mental map" has been coined by Misue et al. [MELS95]. They discuss three concepts on mental models: Orthogonality, proximity and topology.

Based on these ideas, Purchase presented an empirical study that strengthens the assumptions on the relevance for the understanding of node-link diagrams [PHG07]. Depending on the task, preserving the mental map seems to be more complicated than earlier works indicated [PS08]. In a recent evaluation [AP13], the authors showed the effects of map stability to navigation tasks for animated and transitions and for small multiples.

We consider the layout stability as a means to augment recall on recently visited regions of the graph. The results of Marriott et al. [MPWG12] suggest that layout features have different cognitive impact, e.g. favoring symmetry or orthogonality. In an earlier study, Purchase and Samra [SP08] note that minimal node movement may not be the most relevant criterion for mental map preservation. Archambault et al. [APP11] compare animation approaches to small-multiple approaches, but their effect on mental map preservation are inconclusive. We have to note that especially recall experiments are naturally limited to small graphs - and schemes to transfer results to real world graphs have yet to be devised. According to Purchase et al. [PS08], one important criterion is that the movement during the transition from one view to another should be minimized. Also, different quality objectives for node movement have been introduced to enable the user to track the changes between two consecutive frames [FT08]. The shape matching approach we present pays respect to these findings: with the exception of stitched nodes, all nodes move uniformly during the transition process.

While most of these concepts have been developed for the layout of dynamic graphs, we conclude that they also can be applied to extracted, dynamic subgraphs of large static graphs. For both cases the same requirement applies: A significant portion of the visible graph is topologically stable between any two changes. Different metrics on how to measure and optimize these changes has been presented earlier by Branke [Bra01], but most of them compare only key frames of the full graphs. Navigation in partial graph views creates a series of subgraphs that are derived from a larger graph. Diehl et al. presented a strategy for the transition between these subgraphs, but it requires that the navigation path must be known in advance. The subgraphs required for the animation are then derived from that supergraph [DGK01]. However, this approach cannot be adopted for interactive browsing, since user interaction cannot be aggregated beforehand. Lee et al. apply quality measurements to an optimization framework based on simulated annealing [LLY06]. While this approach is generic, the given performance suggests that simulated annealing might not be suitable to support interactive browsing and real-time feedback.

Osawa combines traditional layout based on a mass-spring system with heat models to create stable layouts [Osa01]. The user distributes virtual heat energy to one or more nodes, which is then distributed to neighbor nodes. While this approach seems to be promising, it requires the user to manually tweak the visualization. For graphs with inherent hierarchical structures, a clustering can be built up and used for the layout. This also works for dynamic graphs, as Frishman and Tal show with their work [FT04]. Visual changes in the layout are reduced by adding “spacer” vertices that reserve space for future nodes. In their later work [FT08] they present a drawing algorithm that is tailored for rendering online graphs (i.e. only previous frames are known) efficiently on GPUs. The authors define placement strategies for new nodes and an approach to reduce unnecessary node movement.

3.3.9. Degree of Interest

Some approaches make use of an evaluation function to extract interesting parts of the graph with respect to one or more selected vertices. This idea of a Degree-of-Interest function was brought up by Furnas who proposed to derive the set of most interesting points based on user interaction [Fur86]. *Degree-of-Interest-methods* (DOI) are actually independent of the navigation

concepts mentioned above, but became popular within Focus & Context visualization techniques. In Furnas' initial presentation of the DOI the focus was on the definition of the set of elements that comprise the visible focus rather than the presentation of the focus.

The approach of van Ham and Perer's approach picks up that idea with an adaptation of Furnas's DOI function with respect to graph data [vHP09]. Their work indicates that dynamic graph visualizations can also benefit from that idea. A single node defines the focus and the neighbourhood of this node provides the context. Changing the focus node affects the display of the context. Also related is the approach of Crnovrsanin's et al. [CLWM11], which use the interaction history for the definition of focal nodes. It uses the exploration path of one user or even the paths recommended by multiple users.

In the work of Dwyer et al., fading edges are used to indicate connections to invisible regions [DMS*08]. In a recent extension of their concept of DOI-based exploration of graphs, van Ham and Perer introduce *Graphcues*, which are extensions to visible nodes [vHP12]. Similar to the approach in Section 3.5 they represent links to the interesting but currently invisible parts of the graph. While our approach is based on a signpost metaphor, van Ham and Perer guide the user based on the result of interactive queries. These queries define the interest of individual nodes. As a consequence, the level of interestingness is based on nodes rather than clusters or regions in the graph.

Tominski et al. presented a lens-based technique to visualize large graph data sets [TAvHS06]. The focus acts as a local filter, which suppresses the display of edges unrelated to the visual area, which mitigates overplotting of visual element. A similar filter is used to bring the local neighborhood into the focus. Moscovich et al. [MCH*09] extend this approach with the navigation concept called *Bring & Go*. Guided panning along the connections to neighboring nodes supports navigation in the graph. Their approach is also related to techniques for off-screen visualization.

3.3.10. Off-Screen Visualizations

An *Off-screen visualization* can be considered a variant of Focus & Context, but deserves further elaboration here. Baudisch and Rosenholtz [BR03] compare two visualizations of off-screen locations (isodistance rings and arrows). While the rings performed better on a geographic map, they have no simple equivalent in the complex topology of an arbitrary graph. In fact, off-screen visualization are most often used in the context of geographic maps like the insets techniques presented by Karnick et al. [KCJ*10] and Ghani et al. [GRE11]. One noteworthy exception is presented by Frisch and Dachsel [FD10] who support the navigation in UML diagrams. The position of visual proxies at the screen boundary indicates the global layout and provides context for the orientation. The visualization by Jusufi et al. arranges linked parts of a graph at the border of a focus region that shows one particular part of the network [JKKS12]. Their application is settled in the biochemical network domain.

3.4. Definition of Graph, Context and Focal Node

In mathematical terms, the data we work with has the form of a graph. We define such a graph $G(V, E)$ comprising a set of vertices V together with a set of edges E that connect some pairs of vertices. All of our methods work with both directed and undirected edges without limitations. The techniques in next sections we present are based on the browsing paradigm and can only be applied to connected components. This means that new nodes are reached through their neighbors. If the graph contains parts that are not connected to the rest of the graph, they cannot be reached. We will therefore assume that the graph is connected, i.e. a path exists between every pair of nodes in G . We also define a clustering $C(V)$ as a mapping of V to a set of classes, so that every vertex in G is linked to one or more classes.

Based on the graph G we will extract the focus $F \subset G$, a subgraph, which is defined based on recent user interaction. It is displayed to the user as the visible part of the full graph. It is constructed from the focal nodes $Z \subset V$, a small set of vertices in G and a degree-of-interest function that defines the visible neighborhood of F . The focus nodes are initially selected by the user (e.g. through a textual search query or point and click interaction) and thus are considered to be the most interesting points for the analysis. The subgraph F surrounds the focal nodes and thus provides a context. The definition used here is analogous to the definition in the work of van Ham and Perer [vHP09], but we decided to use the multiple focus points approach as described in Section 3.5 for two reasons:

First, changing a single focus point keeps most of the context as it is induced by the other nodes. Typically, the set of focal nodes is implemented as a first-in-first-out (FIFO) queue with limited size. Every time the analyst puts a new node in focus, it is added to the queue of focus points. It also adds its interesting neighbors to the visible graph. When the queue is full, the least-recently used focus node is dropped from the queue and its neighborhood is removed from the view. Using more than one focus node also keeps changes in the visual structure to a minimum which preserves the context better than the single-focus-version.

The second reason is that it also gives a sense of history which is particularly useful for browsing tasks where the path to the solution is not known beforehand or part of the solution. After the focal nodes have been picked the next step is to derive the context. To achieve that, a degree-of-interest function is used to evaluate the relevance of a node with respect to the current focus.

Regions $R_i \subset G$ are arbitrary subgraphs of G . Because it differs from van Ham and Perer's definition, we want to clarify that when using the word *context* we refer to regions beyond the visible subgraph F , and not the immediate, visible neighbourhood of Z .

3.4.1. Initial Node Selection

The initial pick of focus nodes in a large-scale graph is a non-trivial task. Following the "Overview first, zoom and filter, then details-on-demand" paradigm by Shneiderman we notice that the overview is not available in partial graph visualizations [Shn96]. In order to find potentially interesting regions of the graph, we cannot drill down from a high-level view to a specific region that demands our interest. We have to explicitly perform a search query that iterates over the full graph and select one or more of its results as a starting point. For example, a text search could

provide all nodes whose attributes that match the expression. Alternatively, picking a keyword from a predefined list could select all vertices that are associated with that keyword.

3.4.2. The Degree-of-Interest Function

The concept of a Degree-of-interest (DOI) function is to evaluate the interest of a data element with the respect to the observer. The larger the distance from the current viewport the smaller seems to be the importance for the current task. In the original publication by Furnas, several possible mappings for the parts of this function are presented [Fur86]. For example, the API function can be generated from inherent attributes of the node or its relevance in the structure. The UI function could match a node to a user defined text search query and the distance function can be mapped directly to the minimal weighted or un-weighted distance in a graph. We use a DOI-function as described in Section 3.5 in equation 3.7.

We will briefly sketch how the contextual subgraph can be constructed. A modified version of the Dijkstra algorithm can be used to derive the visible nodes around the focus points. Instead of using a single starting point, this variation works with multiple starting points, namely the focal nodes. The DOI of neighboring nodes is used as edge weight. In every iteration, the node with the highest interest is added to F . Typical stopping criteria are the number of total nodes in the set or a predefined threshold that filters out nodes with a DOI lower than a certain value. As it is, this function might lead to disconnected graphs which are undesirable. A connected subgraph can be created by adding the shortest path between all focal nodes to the visible graph. We will call these visible subsets of the graph *frames* from now on.

3.5. Signposts

In this section we present a local graph view for the exploration of large, static graphs (Contribution C_1). It is a Focus & Context technique that – in contrast to most other techniques – uses a symbolic representation to present the context of the focus region. The focus is the local graph view, and labeled signposts provide cues for the context which is defined as regions of the graph that are outside the visible area. In analogy to real signpost alongside roads, these cues indicate the direction of the shortest path from the visible set of nodes to these invisible or unknown regions.

We provide a definition for the off-screen regions and describe how they are selected based on the visible subgraph. Our approach is partially based on the work of van Ham and Perer, who dynamically extract a subgraph based on an initial focus node and a degree-of-interest function. We picked up this idea to define the set of visible nodes and extended it to support multiple focus nodes. Using the signposts metaphor, regions of a graph that are potentially interesting for the analyst can be hinted with a very small visual footprint. Our approach supports the data scientist in the exploration of large graphs based on a small subset with high relevance and cues towards interesting off-screen areas. We briefly outline a user study we performed to evaluate the effectiveness of the signposts for navigation tasks.

3.5.1. Visual Cues

On the one hand, local graph views must provide enough visual context to support the preservation of the user's mental map. On the other hand, the visual complexity must be kept to a minimum to avoid cluttering which quickly distracts the users focus. A major challenge for such Focus & Context techniques is thus to balance the amount of visual information and the preservation of the user's sense of context.

Over the last couple of years, a broad range of navigation concepts have been discussed to support the user in the exploration of large graphs. Those who use two separate, but linked views at different resolutions or granularities are typically referred to as *Overview & Detail* techniques. This aims to provide the user with both details for close inspection, but also orientation on a coarse scale. Others use the *Pan & Zoom* metaphor of a virtual camera to adjust the visual complexity and the area of interest based on user interaction. In contrast, *Focus & Context* techniques work with a single view, providing a spatial link between the focus area and its environment. Often-used graphical means to accomplish this are distortion techniques such as the fisheye view or magic lenses [SB92,GNBP11]. For a detailed discussion of the different categories, we refer to Section 2.5.

In our approach, graphical and textual cues are combined to refer to the context. In contrast to many related techniques, this context actually lies outside of the current field of view and is therefore not visible.

As the name already inclines, our signposts technique was inspired by real-world signposts that guide travelers towards their destination through unknown areas. In this metaphor, the traveler's visible environment translates to the *focus* area that is visible to the analyst. Graphical cues indicate direction and textual cues represent potential destinations beyond the visible environment. In both cases, signposts provide the *context* to support navigation and exploration tasks.

We transfer this concept for the exploration of static graphs, based on a small visible subset. The focus is depicted by a node-link diagram of a selected subgraph, while the context consists of named glyphs that are attached to certain nodes. They point towards regions that can be reached through these nodes, but lie beyond the boundaries of the visible focus area.

Textual representation bring two advantages: First of all, subgraphs can be represented independently with a very small visual footprint. The second reason is that they can be recalled faster and more accurately than graphical representations [Kos07]. However, the labels might be unknown to the user at the beginning of the analysis and need to be learned before they can be used effectively. Obviously, effective usage requires meaningful labels for the regions of a graph. In some data sets, labels can be naturally derived from data attributes (for example product categories). Many network data repositories include resources like semantic annotations which can be used as labels. This applies in particular for digital libraries. However, the initial cost for the creation of meaningful groups and textual descriptions for unlabeled graphs can be quite high as it typically requires human interaction. We support this manual creation of regions and labels on the fly, but note that this interactive definition process is rather cumbersome for large graphs.

Our approach is primarily influenced by two publications: The definitions of the focus region (see Section 3.5.2) is based on the ideas of van Ham and Perer [vHP09], who transferred the concept of Furnas' degree-of-interest functions to graphs. We use this function to define the

relevant set of nodes from a focus node of interest, but generalize their approach to multiple focus nodes. Our navigation concept is also similar to some extent as it responds to changes of the focus nodes.

The second important publication is the dynamic inset approach by Ghani et al. [GRE11]. It shows insets at the border of the view to depict off-screen graph nodes (see Section 3.5.3). Insets at the boundaries of the focus view refer to connected nodes outside the visible area. The idea behind the insets is to provide the user with relevant information without the need to zoom-out first. Notably, several similar approaches for off-screen visualization exist today [HSS11, JKKS12].

The signpost glyphs in our approach can be compared to these insets, but they point to *regions* rather than individual nodes. In our approach the relevance of a region is derived from its size in terms of nodes and its distance from the viewport. We will show in the following sections, how we generate the contextual information and how we communicate it to the analyst.

In summary, we claim to contribute the following items to the scientific community:

- An extension to van Ham & Perers degree-of-interest function for graphs that supports *multiple focus points*.
- An extension to the generation of the visible subgraph that ensures that it remains connected.
- Visual cues based on direction glyphs and textual cues referring to off screen regions providing context for the navigation.
- Dynamic selection of relevant off-screen regions based on a secondary degree-of-interest function.

In the following, we will show that we do not only contribute the general approach as described, but also make individual contributions to several aspects that we've built on.

The concept we present generalizes the idea of *context* to invisible off-screen regions. They can be referred to with the help of visible signposts. In turn, off-screen visualizations are generalized with regard to the definition of the *viewport*, which is defined by the the graph topology rather than the canvas geometry. Finally, to our knowledge, off-screen visualizations of abstract graphs have not been used in combination with multi-level representations yet.

We consider this approach a Focus & Context technique as defined by Cockburn et al., because this definition does not prescribe the level of abstraction used for the context visualization.

3.5.2. Defining the Local View

In line with the approach of van Ham and Perer, we apply a degree-of-interest (DOI) function to the nodes in the graph in order to identify the most relevant ones for the user [vHP09]. This function assigns an interesting value for every node and those those who cross the relevance threshold are part of the visible subgraph while the others remain hidden. Typically, the areas of high interest are around a certain, user-defined focus point and the interestingness fades with increasing graph-theoretic distance to that node. Consequently, such a DOI function defines the

visible neighborhood around a given focal node. In contrast to the original work, our DOI function supports multiple focal nodes.

The striking advantage of this is the ability to track the development or refinement of the user's interest in terms of exploration history. This is particularly useful for exploratory tasks or where the goal is not known beforehand. In these cases, the path to the solution may be of interest as well.

We will start this section with a detailed description of our extension to van Ham and Perer's DOI function. Then, the derivation of the focus from the set of focal nodes is explained. Applying the DOI function as-is would lead to the construction of subgraphs which contains disconnected neighborhoods. We deal with this problem in the last part of this section.

3.5.2.1. A DOI Function for Multiple Focus Nodes

Van Ham and Perer describe their DOI function as a composition of three individual parts. The first component is called *a-priori interest* (API) which never changes. The second is the *user interest* (UI) which is derived from initial user queries. The *distance-based interest* (D) refers to the distance to a focal node $z \in V$ (see Equation 3.1).

$$DOI(x) = \alpha \cdot API(x) + \beta \cdot UI(x, y) + \gamma \cdot D(x, z) \quad (3.1)$$

We adopt the first two parts, but extend the last component of the DOI function and the way the subgraph F is derived from this function. The definition for our DOI function that supports more than one focal node looks as follows:

We denote the set of focus nodes as $Z = \{z_1, z_2, \dots\} \subset V$ and the distance function $d(x, y)$. In the following, we assume that d is defined for all pairs of nodes in the graph. The distance between nodes that are not connected is defined as infinite. A DOI function should respect that a node that is closer to one or more focal nodes should be considered more interesting than a node away from the focal nodes. The original definition by Furnas [Fur86] of the distance component is as follows (Equation 3.2):

$$D(x) = -d(x, z) \quad (3.2)$$

This function was extended by Heer and Card [HC04] to multiple focal nodes in a hierarchy (see Equation 3.3). However, it emphasizes a convex region defined by the majority of focal nodes which is not suited for multiple focal nodes in a general graph. Nodes that are located outside of this region – even other focal nodes – may be discarded by such a DOI function.

$$D(x) = -d(x, z_1) - d(x, z_2) - d(x, z_3) \dots \quad (3.3)$$

To compensate that weakness, we define an inverse distance vector for every node in the graph instead (shown in Equation 3.5). In this definition, we use the norm of the general L^p function space (named after Henri Lebesgue) that is defined for a real number $p \geq 1$:

$$\|d_Z(x)\|_p = (|d(x)_1|^p + |d(x)_2|^p + \dots + |d(x)_n|^p)^{\frac{1}{p}} \quad (3.4)$$

This norm equals the Euclidean norm for $p = 2$ and the Minkowski-norm (also known *Taxicab* distance) for $p = 1$. For $p < 1$ the DOI distance component is not a metric, since it does not fulfill the triangle inequality. The resulting general definition for multiple focal nodes is defined in Equation 3.6.

$$d_Z(x) = \left(\frac{1}{d(x, z_1) + 1}, \frac{1}{d(x, z_2) + 1}, \dots \right) \quad (3.5)$$

$$D_Z(x) = \|d_Z(x)\|_p \quad (3.6)$$

The definition of the variable p influences how the neighborhoods of the focal nodes are combined: In general, higher values of p lead to the creation of more independent neighborhoods in general.

A *nearest-neighbour* approach can be achieved by using the maximum norm $\|d_Z(x)\|_\infty$, where only the distance to the single nearest focal point is taken into account.

The norm $\|d_Z(x)\|_1$ averages the inverse distance to all focal points. Even though that $p < 1$ is not a valid distance metric, we prefer DOI distance components with values for p in the range $0 < p < 1$ for our approach. The reason is that the resulting distance component emphasizes the area between multiple focal nodes (see the illustration in Figure 3.5). In particular when the connection between the focal nodes is considered at least as important as the set of focal nodes itself, such DOI components prove to be useful. Furthermore, these measures are less likely to split the visible subgraph into separate parts. However, this cannot always be avoided by the DOI function per-se and we describe counter-measures for that in Subsection 3.5.2.3.

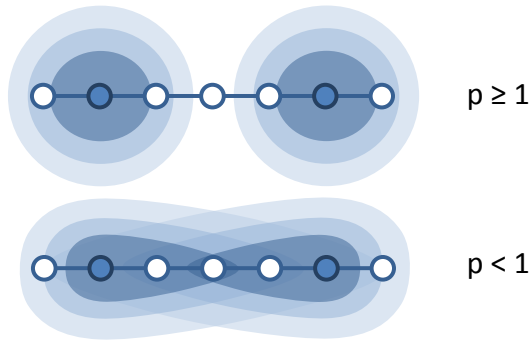


Figure 3.5: Combined iso-distances with two focal nodes (blue) for different values of p . The variable p defines to what extent nodes between two or more focal nodes will be preferred over nodes near a single focal node. High values of p will lead to independent neighborhoods; lower values of p will be more likely to cause neighborhoods to merge.

3.5.2.2. Extract the Visible Subgraph

This subsection will go into the details on how the visible subgraph is extracted from the full data set. Our approach is based on the ideas of van Ham and Perer [vHP09], Heer and Card [HC04] and

Huang et al. [HEW98]. The DOI function is independent from the specification of the focal nodes Z . Typically, these nodes can be selected interactively by clicking on them. Doing so expands the currently subgraph F so that it includes the neighborhood of the selected node. Our approach uses a double-ended queue to track the selection history of the nodes in Z . Old focal nodes drop out of the queue as soon as a predefined maximum size has been reached. Such an approach has also been used by others [HEW98].

When a new focus node is added, its distance to all other nodes is computed with a distance function $d(x, y)$ that is based on the DOI values for each node.

The work of van Ham and Perer suggests a modified version of the *Dijkstra* algorithm that works as follows: Initially, only the focus node is visible. In an iterative process, the DOI values for neighbors are evaluated. The algorithm adds nodes with the highest DOIs until a predefined number of nodes has been added to the visible set. Interestingly, this same method can be applied to multiple seed nodes without change and even without extra computational cost. Multiple focal nodes may share the same neighborhood. While this approach works quite well, it does not guarantee that the resulting visible sub-graph is connected.

3.5.2.3. Connect Disjoint Sub-graphs

Our adoption of this algorithm takes this issue into account. It connects the neighborhoods with chains of nodes along a reasonably short path between them. We refer to these connecting elements as *bridges*.

This is very much in line with the idea of the interactive expansion of the focus area, as the set of visible nodes serves as an anchor for detailed inspection. A user that wants to investigate the area between different focus nodes, the bridges make this area visible and highlight important connections.

First, the algorithm checks if the set of visible nodes forms a single neighborhood. A unique identifier is applied to all focus nodes. This marker is then applied to its neighbors, neighbors of neighbors and so on. If a node has already been marked before with a different ID, the two markers are merged into one. The process stops as soon as all nodes have been marked. In the end, all connected areas are marked with the same ID. The number of distinct IDs consequently indicated the number of disjoint sub-graphs.

In a follow-up process, these separate regions are connected by the aforementioned bridge nodes. Our approach is an iterative process that computes bridge elements for two disconnected graphs. See Figure 3.6 for an illustration. It starts at the smallest sub-graph N_0 performs a growth process that is very similar to the definition of the visible region that was described in the previous section. The difference is that the number of nodes is not limited.

The neighborhood N_0 is increased until a different neighborhood is reached. This is again done using a Dijkstra-based shortest-path search that directly gives the optimal path between the two regions. Adding the nodes along this path to the set of visible nodes naturally connects them. The process as a whole is repeated until only one neighborhood is left. The resulting single area defines the focus F , which is the visible subgraph that is used in the visualization.

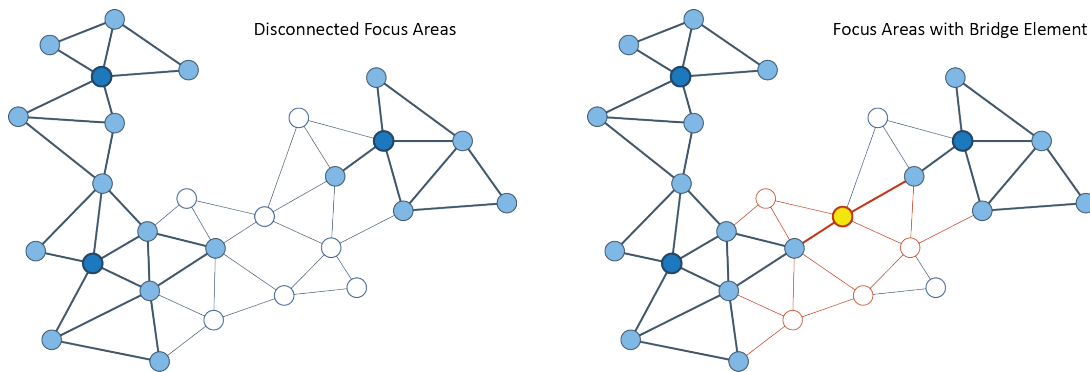


Figure 3.6: Nodes for the focus are selected by a two-step process. First (a), the area around the focus nodes (1,2 and 3) is increased to a pre-defined number of nodes. The DOI function defines which nodes are preferred. During this process, different neighborhoods can grow together (1 and 2, left). Second (b), disjoint areas of the visible graph are connected using bridge elements. Starting with the smallest disconnected sub-graph (3, right), nodes are added until a different, visible part of the graph is reached. The bridge (B) is created along the shortest path as found by Dijkstra's algorithm.

3.5.3. Signpost-based Context

We described how the *focus* is defined in the previous section. This focus area $F \subset G$ allows for the detailed inspection of a subgraph of interest. It also enables the user to interactively explore adjacent regions by changing the focus nodes. In this visualization, the *context* consists of glyphs which resemble signposts (see Figure 3.7).

These markers are located at the boundaries of the visible area. In analogy to real-world signposts, they direct to areas of the graph which could be of interest to the user, but lie outside the visible focus area. By selecting different focus nodes the visible subset changes accordingly and the user can thus navigate through the graph.

The purpose of the signpost glyphs is to provide context information on the current location of the visible subgraph in relation to other regions of the graph. Whenever the focus changes, the visible sub-graph changes and consequently the the selection, position and direction of the glyphs are updated.

We will now describe how these signposts are created and start with the definition of the off-screen regions of a graph that can be referred to with the help of the sign glyphs. Then, will explain how we select the subset of these regions that will be used in the visualization, based on the current focus. In the last step, we will explain how the signposts are laid out on the screen.

3.5.3.1. Region Labels

A signpost often points to a far-away places such as cities, a regions or even different countries. When it refers to large distances, it will probably not refer to an individual building or street. A signpost that points to closer targets may refer to smaller anchor points as it is expected that the

traveler is – to some extent – familiar with the closer environment. Naturally, the label of the long-distance signpost implicitly comprises all minor signposts within the boundary of its target area.

We transfer this concept to graphs by aggregating nodes to *labelled regions* and define a *region* as a connected subgraph $R \subset G$. The regions in the data set do not need to partition the graph data, i.e. they can overlap and even form hierarchies. This can be used, for example, to realize different levels of detail (see Section 3.5.3.2). In general, there are no limitations with respect to the definition of the set of regions $\mathcal{R} = \{R_1, R_2, \dots\}$.

Human-readable labels are required to make signposts meaningful. We will assume from now on that this is the case for each region in the data set. The labeling procedure itself is not part of our contribution.

The method we present can be used for navigation in graphs that is enriched with such metadata. Today, an ever-growing number of open-access repositories provide such graph data, often even annotated with semantic information. Such data can be used to derive representative labels for groups of similar elements. A prominent example is the *Resource Description Framework* (RDF) data type that is used in the Linked Open-Data Community. Potentially useful attributes can be *Product Categories*, *Authors*, *Topics* to define region labels. In our case study, we will use the *ICD-Codes* that are defined for groups of similar diseases (see Section 3.5.4). An attribute defines a region as the set of nodes with the same value. Thus, even nodes of different types may be in the same region if they share attributes with identical semantics.

A straight-forward approach would be to directly use a single data attribute to define the regions. Naturally, this attribute can then be directly used as a label. Most nominal attributes in the data are candidates to define regions. Some data sets are annotated with a taxonomy so that the inherent hierarchy can be exploited to create regions with different sizes depending on the desired level of detail. Regions that belong to the same super-category can be merged and labelled with the name of this class.

Despite the fact that there are no technical restrictions on the definition of regions, some attributes are better suited than others. A potential problem when using data attributes is that nodes with that attribute do not necessarily form a connected group in the graph topology.

In order to be able to point towards a direction, the region of interest should be sufficiently *compact*. It not need to be connected, but the average distances within the region should be significantly smaller than the average distances of the complete graph. Only if this requirement is fulfilled, region can be reasonably represented with signposts. Note that this requirement is highly similar to the definition of graph *clusters*.

Regions can also be generated automatically with graph clustering algorithms. These region often represent the graph topology better than manually creates ones, but they do not have a meaningful label per-se. In a second cycle, classification methods could be used label the generated regions automatically, but this is beyond the scope of this work. We approach the problem from the opposite direction by enabling the data analyst to modify and extend the set of regions. Once a a set of nodes appears to be relevant enough to return to it later, they can be selected with a lasso tool. Selected nodes will be stored in a new region with a user-defined label (see Figure 3.7). The region is later referred to with its own signpost as soon as it drops out of the visible focus.

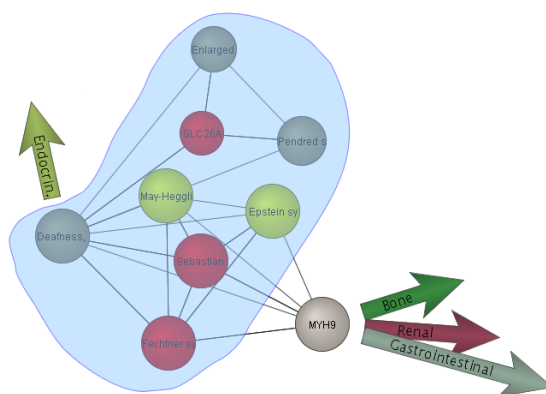


Figure 3.7: Two signposts guide the user towards other regions in the graph. They point along the shortest path to those regions. The user selected a group of nodes that should form a new region with user-defined label.

3.5.3.2. Filtering Context Regions

Large graphs can contain dozens or even hundreds of regions. Without pre-defined limitation on the definition of regions, the number of potentially relevant regions can be very high as well. Consequently, the number of visual signposts must be filtered to a sensible amount in order to avoid visual clutter and overplotting in the display. In analogy to the definition of visible nodes for the focus area, we apply a degree-of-interest scheme to filter the context. We therefore assign a DOI value for every region of the graph. This value depends on specific properties of the region such as its size and compactness.

However, it also depends on distance from the current set of nodes $F \subset G$ in the focus. Whenever the focus area changes, regions may become more or less relevant in the context. As a consequence, signpost glyphs appear or disappear accordingly.

Formally speaking, our definition of the degree-of-interest function for graph regions is based on the following properties:

- *Intersection with the focus:* if a region R intersects with the focus F then the region is automatically excluded from the selection.
- *Region size $|R|$:* bigger regions are considered more important than smaller regions. The practical reason for this scheme is that the whole graph can be covered more easily by referring to a few large regions.
- *Region distance from the focus $d(R, F)$:* regions that are close to the focus get a higher relevance score. It is assumed that they are more closely related to the nodes in the focus, i.e. the part of the graph the user is currently exploring (see Section 3.5.3.3).
- *A-priori interest $API(R)$ and user interest $UI(R)$:* these are highly similar or even equal to those that were defined for individual nodes (see Section 3.5.2). For example, the a-priori

interest for user-defined regions should be increased, because the user explicitly expressed interest in these regions by creating them.

- *Exclusion:* in some cases, regions can have significant overlap even when defined with different methods. While this is interesting from an analytical point of view, having multiple signposts referring to the multiple regions which differ only slightly does not support the user's mental map much. We therefore remove regions that are highly similar to any other region in the selected set.

To fulfill the first requirement, regions which intersect the focus are assigned with a DOI value of zero. We define the DOI function for the regions based on these requirements as follows:

$$\widehat{DOI}(R) = \alpha \cdot API(R) + \beta \cdot UI(R) + \gamma \cdot D(d(R, F), |R|) \quad (3.7)$$

Function $D(\cdot)$ reflects the fact that region size and distance from F both influence the DOI value. We will briefly sketch how the two factors can be combined based on a small example scenario: a large region in the graph which overlaps a set of smaller regions.

From a distance, the large region is generally more interesting than the smaller ones. This high-level is enough to provide some orientation. If the user is interested and moves the focus closer to these regions, we can assume that a higher level of detail – i.e. more information is desired. Consequently, smaller regions should be preferred at close range to support this scheme.

The class of functions D in the following equation satisfy this property:

$$D(d(R, F), |R|) = \frac{\omega}{d(R, F)^\omega} \quad (3.8)$$

$$\omega = 2 - \frac{\log(|R|)}{\log(|R_{max}|)}$$

The largest region is denoted by constant R_{max} . Finally, we need to define how we ensure that all regions in the selected set are sufficiently different from each other. Our approach is based on a list of excluding regions X_1, X_2, \dots that are computed for each region $R \in \mathcal{R}$ beforehand. These mutual exclusions can be based on the number of common nodes, for example.

We modify the DOI based on this last component and obtain the final equation as follows:

$$DOI(R) = \begin{cases} 0 & \text{if } R \cap F \text{ is not empty,} \\ 0 & \text{if } \max_i(\widehat{DOI}(X_i)) > \widehat{DOI}(R), \\ \widehat{DOI}(R) & \text{otherwise.} \end{cases} \quad (3.9)$$

3.5.3.3. Defining Region Distances

The graph of regions \mathcal{H} is defined as a weighted graph. Its nodes are the regions \mathcal{R} of G , its edges define adjacent, intersecting and nested regions.

There is no natural distance measure for regions, as they are defined as sets of nodes. A variety of approaches exist to derive distances, many of them are used in agglomerative clustering. The work of Berkhin [Ber02] gives an overview on the topic and surveys related work. In order to

support graph navigation an approximate solution which enables a reference to the right direction suffices. We therefore chose the graph-theoretic distance between regions to define the weight of the edges.

The minimum distance between a pair of nodes in different regions is called *Single linkage* and the easiest one to compute. However, there are two problems with this distance measure. The first one is that distances *within* regions are ignored. As a consequence, distances between two regions in G are underestimated in general. Second, it cannot be applied to intersecting and nested regions – the single-linkage distance is zero in both cases. We chose the *average-linkage* distance for the definition of \mathcal{H} to overcome these limitations. The graph \mathcal{H} is a coarse representation of G and can be exploited to speed up the distance computation. This high-level abstraction of the graph can be pre-computed.

Every time the focus is changed, the relevant regions and their distances are recomputed. With increasing size of the graph, this can become a time-consuming operation. However, the majority of the regions are known in advance and their pairwise distances can be pre-computed. Whenever the focus area is updated, only the distances to the closest regions need to be updated (see next section). Distance to regions that are further away can be derived from the distance to the closer ones and a look-up in the pre-computed distance table for the region of interest (see Figure 3.8 for an illustration of the approach). We note that this computation operates both on the graph of nodes G and on the graph of regions \mathcal{H} .

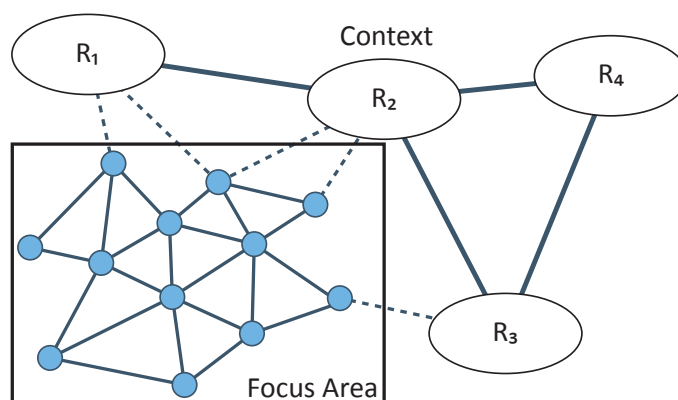


Figure 3.8: The visible focus area is surrounded by a number of invisible regions. The distance to all regions is recomputed whenever the focus changes. However, only distances to neighboring regions (R_1, R_2, R_3) are updated. The distances to more distant regions (R_4) are based on the close ones plus the precomputed distance between the regions.

3.5.3.4. Glyph Placement

The glyphs in our approach convey at least two pieces of information. The first one is the label of the region that the signposts points at. The second one is the visible node, which can be expanded in order to move along the shortest path towards this region.

Computing the distance between two regions and computing the distance to the closest visible node are two highly correlated problems. It is therefore not surprising that both can be solved using similar approaches. We first compute the distance to the closest regions in G , based on the modified *Dijkstra spanning-tree* algorithm in a similar way as we did for finding the neighborhoods nodes in the focus area.

The entire set of visible nodes (i.e. the focus F) serves as seed nodes for the algorithm. Every time a node on a shorter path is encountered, the visible node it has been reached from is updated as well as new minimal distance for the regions of this node. As soon as a fixed number of nodes have been visited, this process stops. We define the regions that were found in this process, *close* regions.

The same algorithm can then be used to compute the distances to regions (in the graph \mathcal{H}) that are further away. The previously defined set of *close* regions serves as now the seed nodes for the algorithm. Summing up the distances of the two computations gives the final distance of all (distant) regions.

As soon as this is complete, the shortest distance to F is known for every region in the graph. Also, the visible nodes it can be reached from along the shortest path are known. In general, the glyphs are sorted by decreasing interest with respect to their associated regions.

When two or more visible nodes share the same interestingness value for a region, the following strategy is used to find the best match. In case the node was used in the previous interaction cycle, the glyph will be attached to this one to keep it in place. This minimizes visual changes across focus changes. If the signpost was not visible before we choose the candidate node which has fewer attached glyphs to reduce overplotting. If there are still multiple candidates after this, the node is picked randomly.

The last thing that is missing is the direction of the signpost glyph. In our approach, it points along the shortest path towards the labelled region. This first edge connects a visible start node in the focus region with an invisible one that lies just outside the focus.

For static layout, the position of this node is known in advance and the edge directed can be derived. Dynamic layouts typically do not consider invisible nodes and edges in the layout. In order to derive the edge direction, we add all direct neighbors of the focus area F to the dynamic layout. No other changes are necessary and our approach therefore works seamlessly with different dynamic layout methods. Those dynamic layout that support animated transitions, the glyph will follow the transition of its (virtual) edge.

3.5.4. User Study

We tested our approach in an initial experiment that compared multi-focal graph exploration with and without signposts. We wanted to test whether the use of signposts supports the user in the navigation and orientation in large graphs.

We recruited 13 participants aged between 21 and 30 for the experiment. All of the participants were graduate or undergraduate students in computer science or computer-science-related courses. Our experiment was a *within-group* test. The experiment consisted of two almost identical tasks, each performed under different conditions. To filter out undesired learning effects we alternated the sequence of the tasks performed by each participant.

In each task, participants were asked to find a pair of nodes, one each from two predefined categories, and a path connecting the two nodes. We selected the categories such that the tasks were easily solvable with the available interface. For the test we used a precalculated Fruchterman-Reingold layout. In both tasks five focal node changes were necessary to reach the target. The number of focal nodes to define the focus view was limited to three. The start conditions for each task were identical. The only difference between the tasks was the availability of signposts, providing additional contextual information to the participants.

Possible user interactions included panning and zooming, browsing the graph by adding a node to the focal node history and the selection of nodes for detailed information. Focal nodes were highlighted with red boundaries. The graph that we used for the experiment was based on data from *The Human Disease Network* by Goh et al [GCV*07]. We extracted the largest connected subgraph, because in its present form, our technique cannot be applied to disconnected graphs. We decided to use this particular data set for two reasons. Firstly, each node was already assigned to one of 23 different categories. Thus we had a predefined set of regions to use for the definition of signposts. Secondly, the meaning of the nodes (either genes or diseases), the links between them and the categories that grouped them were clear to non-experts without the need for detailed explanation. To ensure an equal level of expertise among participants we made sure that all participants had little or no knowledge of this particular data set or medicine in general.

The nodes in the original graph were either diseases, or genes linked to those diseases. We did not distinguish between diseases and genes in our visualization, since the same categorization was applied to both types. We converted the originally directed graph into an undirected graph and filtered out gene-gene relations to reduce its density. The final graph contained roughly 1500 nodes and 5500 edges.

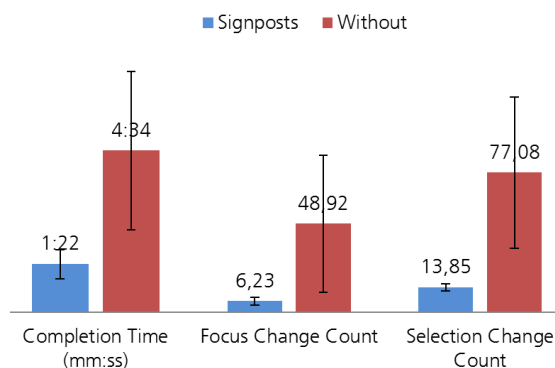


Figure 3.9: *The results of the experiment: lower mean and lower standard deviation in completion time, focus change count and selection count for the visualization with signposts.*

To test our method, we measured the task completion time, as well as two click-based statistics. The time was measured between the moment the task was given to the participant and the moment the participant he or she had found a path and reported that to us. The click-based measurements were the number of focal node changes and the number of selection clicks (to view detailed information about the nodes). Figure 3.9 shows the average completion times for both tasks and

both conditions. Direct comparison reveals that the testers were faster with the visualization that contained landmarks.

The difference in the average task completion time, the number of focus changes and selection clicks points to an improvement in graph navigation through the use of signposts.

3.5.5. Discussion

The user study was conducted to gain a better understanding of the problems users encounter when using our visualization approach. One of the major problems of the participants was that they forgot where they had started and which part of the graph they had already explored.

The queue of focal nodes we used has a length of maximum three entries. Some participants suggested having a larger number of focal nodes to better support tracking the exploration history. However, a larger number of focal nodes caused the edges to be cluttered in the static layout. Similarly, other suggestions included explicitly storing the entire history of focus nodes to document the exploration path. We expect that multiple focus points are likely to work better in conjunction with dynamic layout techniques.

The idea of having signposts was understood without explicit explanation and recognized as helpful by the participants. For the study, we presented a maximum of three most interesting regions per node to avoid visual cluttering. This sometimes leads to the problem that the region that was searched for was not always visible. One way to overcome this problem would be to interactively browse through all signposts on demand – similar to browsing through a stack of cards. A more sophisticated way would be to track the path the user follows and explicitly highlight the next steps along this path.

As discussed in the previous sections, an important requirement of our approach is the definition of meaningful labels. For large graphs, it is sheer impossible to create them manually and automated methods might fail to generate meaningful groups and labels for the groups. However, based on the approach of user-generated cluster labels, we see two interesting alternatives for future work. The first one is about interactive refinement of automatically generated regions, the second one is a crowd-sourcing approach to create a public repository of user-defined regions for large graphs.

In this chapter we presented a new approach to navigation in large graphs that is based on Ham and Perer's idea of presenting a meaningful sub-graph around a single focus node. Our approach goes beyond the original work in two aspects. It supports multiple focus nodes and deal with the problems that come with this. Additionally, it also provides visual cues to other regions of the graph that could be worth exploring. The cues also indicate the direction of the regions in order to support the user in navigating towards them. However, the size and number of displayed arrows is difficult to adjust a-priori. Currently, only up to three arrows are displayed next to each other per edge to avoid overplotting. If the shortest path to more than three regions leads through the same node, the glyphs are rendered on top of each other, similar to a stack of cards. This indicates that there are more glyphs underneath, but additional interaction such as browsing is required to make them readable.

The signposts idea offers the potential to show more information than the region name. Size, length, color or transparency could be used to represent more information about the region or the

path leading to it. In accordance to real-world signposts, the distance to the target regions could be added to better support the user's orientation. Apart from showing information, the signposts could also be used for interaction, for example to short-cut the path to the selected region.

The implementation that was used for this approach and study is based a single, static layout to ensure that the position of nodes in space does not change over time and thus confuse the user. A dynamic layout applied only to the visible nodes would be significantly faster and needs to deal with fewer constraints as the number of visible nodes is a lot smaller. Nodes that become visible through a change of focus would be added to the layout on the fly. However, the initial placement of new nodes and edges in an existing layout is not trivial as it should not impair the existing layout. We refer to Section 3.6 for a detailed approach and additional discussion.

3.6. Stabilized Layouts

In this section we present a set of extension techniques to stabilize interactive dynamic graph layout algorithms (Contribution C₂). It works with different existing Focus & Context methods. We first deal with the initial placement of newly inserted nodes to mitigate acting forces in the layout algorithm. Then, their influence on the existing layout is gradually increased to create a smooth transition between the old and the new layout. To complement this approach we use a look-ahead strategy that integrates additional nodes in the layout to stabilize the layout even more. Our approach is validated using both quantitative and qualitative measures.

3.6.1. Damping Node Movement

While the DOI function that was defined by van Ham and Perer [vHP09] is based on a single focus point, we extended the function to work with multiple focus points in Section 3.5. The layout of the displayed subgraph that is used in local layout techniques plays a key role. However, the approach represented in section 3.5 does not yet consider smooth transitions between the different extracted subgraphs. We claim to contribute a combination of features that improve the user experience with focus-based navigation concepts, based on the ideas that were presented in the previous section:

- A new placement algorithm that puts new nodes close to their neighbors so that acting forces are kept minimal.
- A weight function that integrates new nodes in the existing layout in a smooth transition by gradually increasing their mass.
- A layout looking ahead for nodes that potentially visible after the next focus node change.

These features can be used as separate improvements or in an integrated system. As a result, nodes are added and removed smoothly and node movement is reduced in general. This keeps the visual changes in the existing layout to a minimum. We think that their effect on the layout reduces the cognitive effort for the user and thus support the preservation of the mental map.

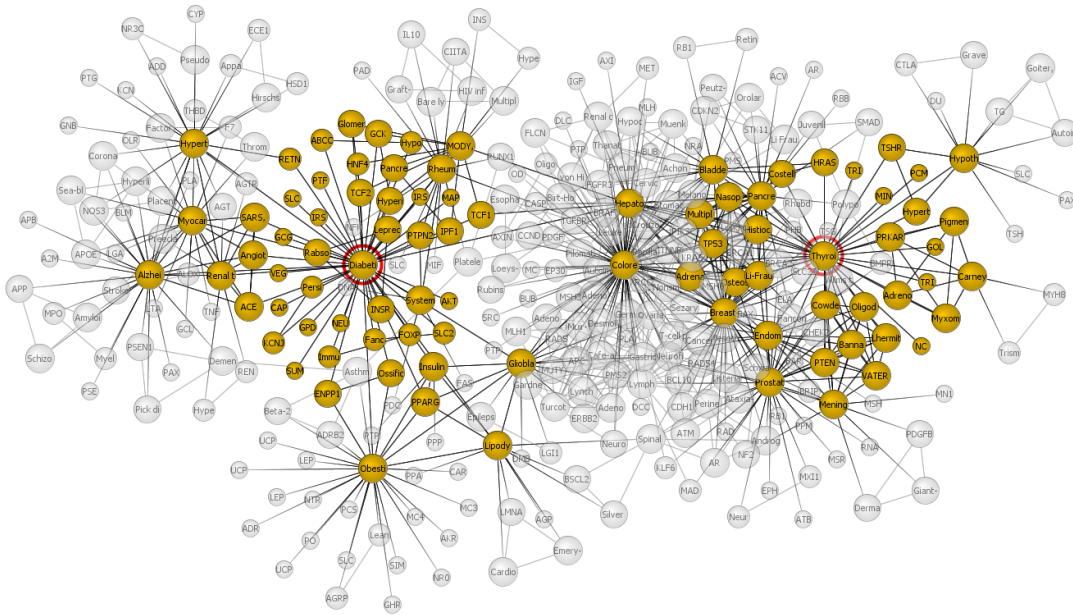


Figure 3.10: A screenshot from the running system showing a partial view of a graph including a set of focal nodes (marked with circles) and a set of visible context nodes. Also, a number of additional, potentially interesting ghost nodes (semi-transparent) are integrated to stabilize the layout, but not shown to the user.

We employ a classic force-directed layout algorithm based on the mass-spring-model that is described by Fruchterman and Reingold [FR91]. Its main actors are spring tension and repulsion forces. For the initial set of nodes no placement constraints exist. Also, for subgraphs that are not connected to the rest, random placement can be used. All other nodes are at least constrained by edge forces that act between neighbors.

3.6.2. Placing Nodes

A typical problem of some graph layout techniques is the at least partially random behavior caused by a random initialization of node positions. The key idea here is to avoid sudden changes in the visualization through reduction of acting forces. Selecting a visible node as focus point creates a new, different set of context nodes. In order to keep the layout stable, new nodes should be placed with a distance equal to the rest length of the edge to satisfy spring constraints. However, it also desirable to place nodes as far away from all other nodes as possible to make use of free space and avoid repulsion forces.

The first aspect can be easily satisfied if the new node has only one edge. It can be placed anywhere on a circle with a radius equal to the desired edge length around the existing node. If two distance constraints exist the range of valid positions is reduced from a full circle to two points on that circle (see the left image in Figure 3.11).

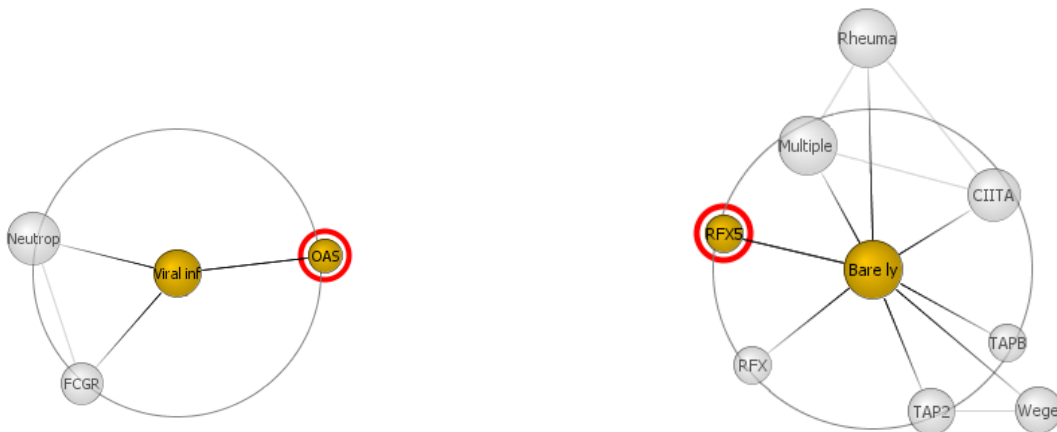


Figure 3.11: Up to two connected nodes (semi-transparent) can be placed on the circle around the existing vertex. Additional nodes are placed on a circle around the center of gravity of already inserted nodes

For nodes that have links to more than two neighbors, no general solutions exist that fulfill all criteria. We thus have to approximate the ideal position. First, the center of gravity of all neighbor nodes that are already in the layout is computed. Then, the final node position is a position on the circle around the center of gravity with a radius of the desired edge length (see the right image in Figure 3.11). This node potentially lies outside the previously mentioned circle and does not fulfill any of the imposed constraints, but it is pushed away from the areas with high node density.

The placement on the circle is performed with respect to the number of nodes in its proximity. This area-based density measure is also required for the computation of repulsion forces of the Fruchterman-Reingold algorithm. Often-used methods to find empty space and perform distance tests quickly are spatial hashing algorithms. They define a grid-like structure of buckets that contain the graph vertices based on their position. Nodes are added and removed from the buckets as the move over the virtual grid. Given a point location the algorithm retrieves its corresponding bucket (based on hash keys). All other nodes in the bucket are considered to be close enough for further distance testing. Depending on the search distance, adjacent buckets are inspected, too.

We can make use of this data structure and query different positions on the circle. We count the number of nodes in the search area of every query location and choose the one with the lowest hit count. This ensures that new nodes are placed where most space is available.

Vertices that fall out of scope are removed from the visible graph. Typically, these nodes have been in the layout for a while and thus have “settled”. The forces that act on them are rather small. Consequently, removing such a node hardly affects acting forces on other nodes and can thus be removed without having a strong visual impact.

3.6.2.1. Look-ahead for Invisible Neighbors

When new nodes are inserted in the layout and displayed immediately, they typically move a lot in order to satisfy all repulsion and edge forces. This can be mitigated by meaningful placement of new nodes but not cured entirely.

This is why we propose a look-ahead technique that also adds nodes to the layout that might become visible, if one of the currently visible nodes was selected as a focus point. Every time a node is added to the visible graph, the DOI function is evaluated for its neighbors as if the node was in the queue of focal nodes.

All nodes that would then become visible have already been added to the layout subgraph before. However, this subgraph can quickly become quite large, especially for highly connected graphs (see Figure 3.12). A restrictive DOI function that filters out the majority of neighboring nodes mitigates the problem.

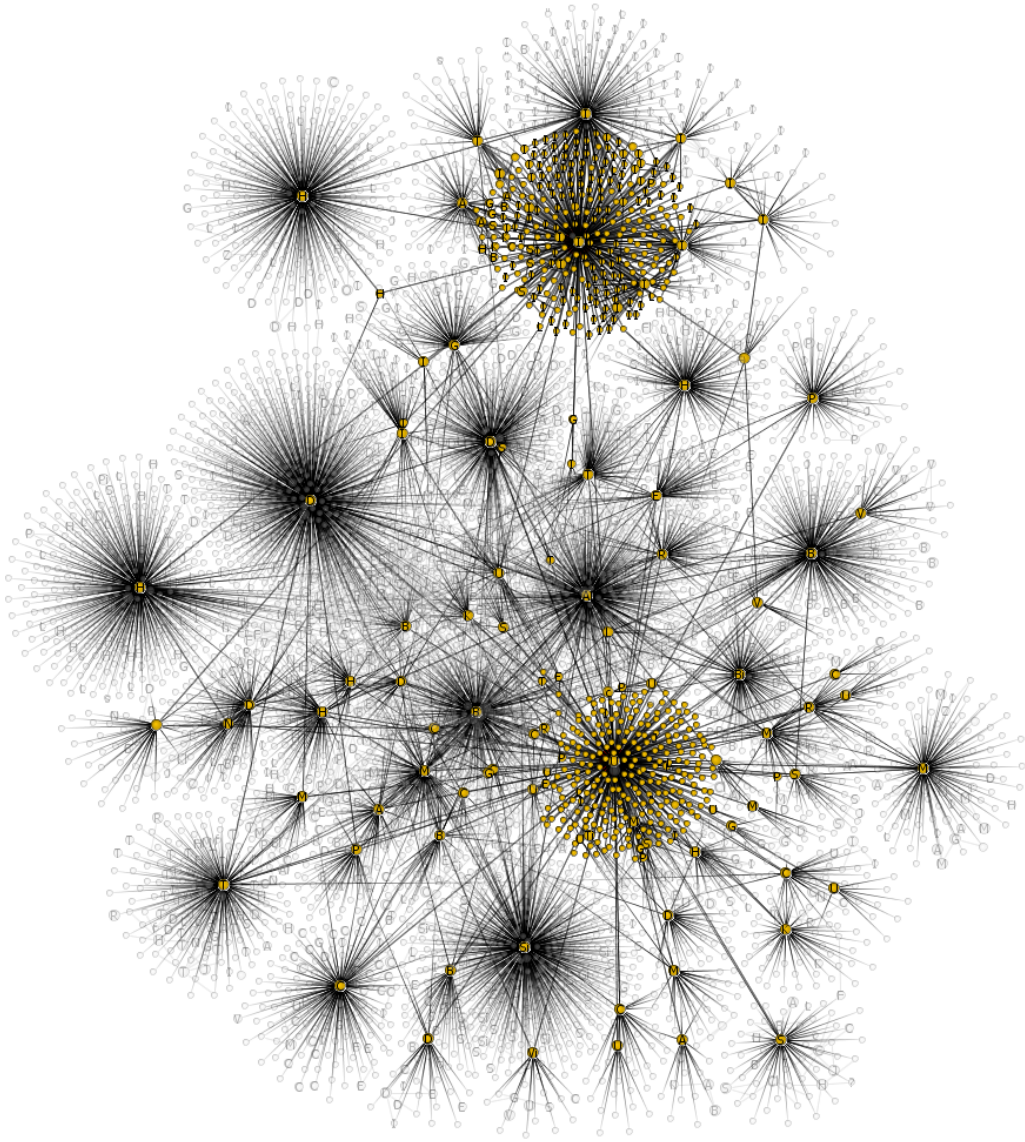


Figure 3.12: *A small subset of a graph with many potentially visible neighbors can have a significant impact on the layout. In this illustration, only one single node has been selected. This star is one of the most central nodes of the cluster and linked to a large number of other stars. As a result the neighborhood is very large even for small hop distances.*

3.6.2.2. Gradually Increasing Node Weights

When many new nodes are placed around an existing node, the existing node will jitter heavily until the newly inserted spring forces have relaxed. We thus modify the algorithm by adjusting the mass of vertices over time. The goal of this modification is to adjust the inertia of nodes, depending on the time spent in the user’s focus. Following the “separation of concerns”-pattern we first create a function that provides the number of layout iterations for every node. We track this by counting the write operations in the layout on a per node basis. Removing a node sets its counter back to zero.

We then derive the node weight from the normalized values of the counting function. It is desirable that invisible nodes reach the mass of visible nodes smoothly as the mass of a node is in direct relation with the acting forces that are used in the layout algorithm. For two vertices with mass m_1 and m_2 the distribution of forces can be computed as:

$$f_1 = \frac{m_2}{m_1 + m_2} \quad f_2 = \frac{m_1}{m_1 + m_2} \quad (3.10)$$

The larger the mass of a particle is, the weaker is the influence of the force and the stronger is the influence on its counterpart. The two functions are plotted in Figure 3.13.

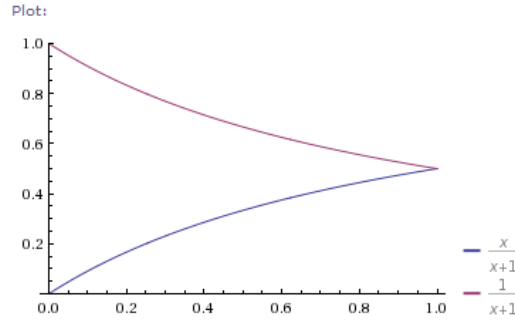


Figure 3.13: *The weight function starts with full effect on the newly inserted node. The influence is shared as the number of layout steps increases until equilibrium is reached and both nodes are equally affected.*

A similar idea has been presented by Huang et al. who introduce friction forces to stabilize dynamic layouts [HE98]. The longer a node is visible the large its friction coefficient gets.

3.6.3. Test Results

The graph data that is used for the tests has been taken from the Diseasesome project. We extracted the largest connected subgraph (disconnected parts cannot be reached with exploratory search) which contained about 1500 nodes and 5500 edges. Although the edges per node ratio has an influence, our approach works on partial views only and is thus independent of the graph size per se. Starting at a randomly chosen focus node, we explored its neighborhood over six focus changes, keeping all explored nodes visible. This created a sequence of six keyframe layouts with

an increasing number of visible nodes (from 4 to 69). A transition between any two keyframes is calculated with 100 iterations of the simulation.

To test our methods we measured the node velocity of all visible nodes in three different settings. Few nodes with high velocities should implicate a larger penalty than many slow movements. This is why we measure the average of squared velocities.

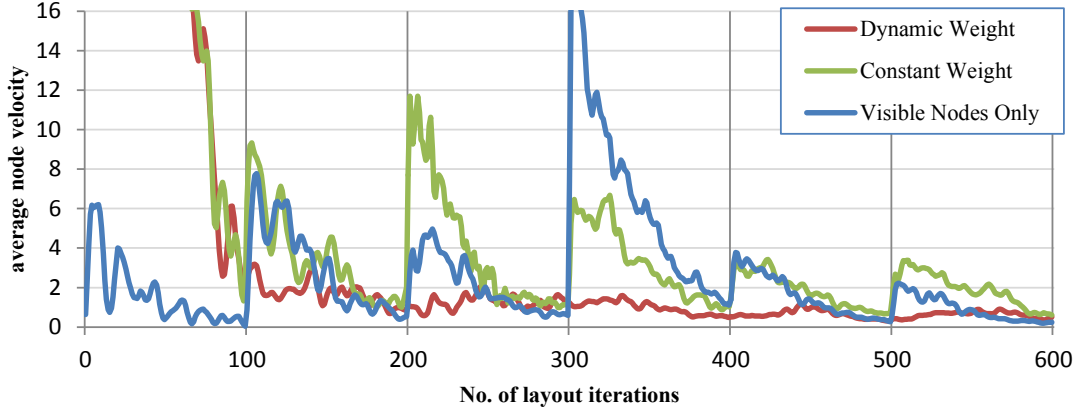


Figure 3.14: Node velocities for three different setups measured over six transitions with 100 layout steps each. Using only visible nodes has the highest average node velocity (blue). Including invisible neighbors reduces velocity peaks (green). Using dynamic node weights reduces the peaks even more as the influence of invisible nodes is still very small after their insertion (red).

The first measurement (blue) was performed with constant node weight and without the invisible neighbor layout. It performs best during the first frame as only 4 nodes need to be laid out. However, the fourth subgraph contains a star node that introduces a large number of new nodes which causes the peak at iteration 300. The second test run (green) also included the layout of the invisible neighborhood. The first subgraph contained additional 15 nodes that exert forces on the visible 4 nodes causing a rather poor performance for the first 100 iterations. At the start of frame 3 (iteration 200) a fair amount of nodes that could become visible in frame 4 has already been added. This causes a similar peak as in the first run, but less high and shifted by one keyframe. The peak at the begin of the forth frame is thus significantly smaller. The gradual increase of node weights has also been activated for the third test run (red). As before, the average velocity for the 4 nodes in the initial frame is very high, but once they have settled down, it outperforms the other methods, especially when a large number of nodes is added simultaneously.

To validate the generality of our presented approach we conducted a second run of tests with a different data set. Using the public API of a large online retailer we were able to access its enormous product database. The network that was used for our test contains roughly 550'000 entities linked by a binary 'is-similar' relationship. Roughly 1.2 million of those links were used to form the edges of the graph. Most of the listed products are associated with at least one category they belong to. This information was used to annotate nodes that are deemed to be important links towards interesting parts of the graph. Details on these visual cues can be found in Section 3.5.

To demonstrate the usefulness of the proposed approach we explored the graph using a dynamic view and recorded the generated animation. A set of selected still images was extracted and is presented in Figure 3.15–3.18.

The first series of images illustrates the expansion of the gray “Be” node at the upper right corner. Before being selected, all directly connected neighbors are already layed out properly (see Figure 3.15 left, transparent nodes). After the selection event the previously invisible nodes appear. Now, using the look-ahead strategy the neighbors of the newly appearing nodes are inserted into the layout (see Figure 3.15 center, transparent nodes). While still being invisible, the algorithm computes an optimal layout for these nodes in case the user explores in this direction which would make them visible.

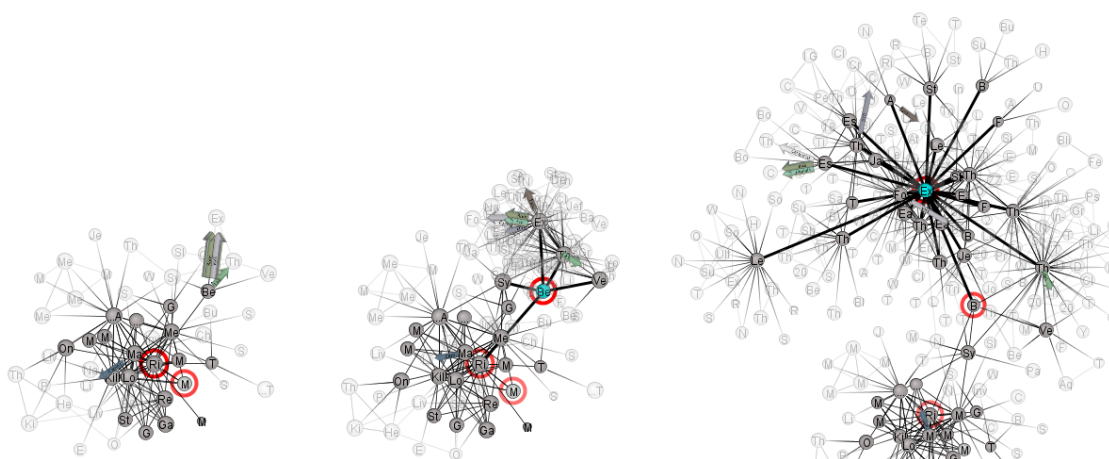


Figure 3.15: Exploration sequence in the product similarity graph. Note that the semi-transparent ghost nodes are not visible in the visualization, but exert forces. Left: original layout. Center: the node “Be” is selected for exploration. Its neighbors become visible and neighbors of neighbors are inserted into the layout without being visible for the user. Right: over time, the layout settles when a locally optimal solution has been found for all nodes.

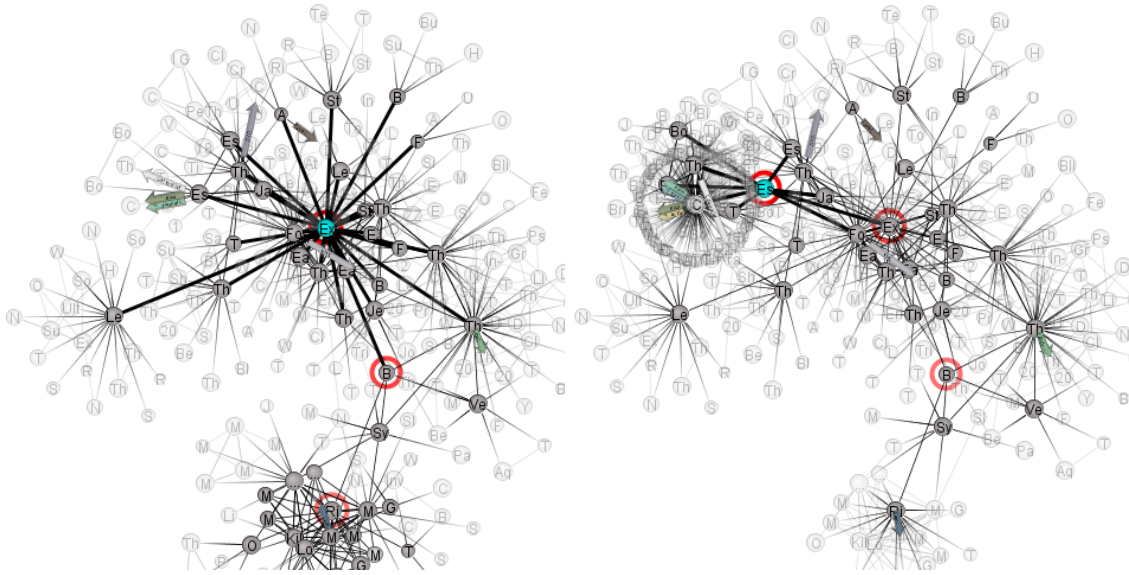


Figure 3.16: Exploring denser region of the graph shows how the node placement algorithm works. At the left side of the visible subgraph, the “Es” node is selected for exploration. This triggers the visibility of the star node “C” (right image). Adding all neighbors leads to a prominent ring around the node, indicating that most neighbors are only linked with “C”. Those who also have other neighbors are put in the center of gravity which is also close-by.

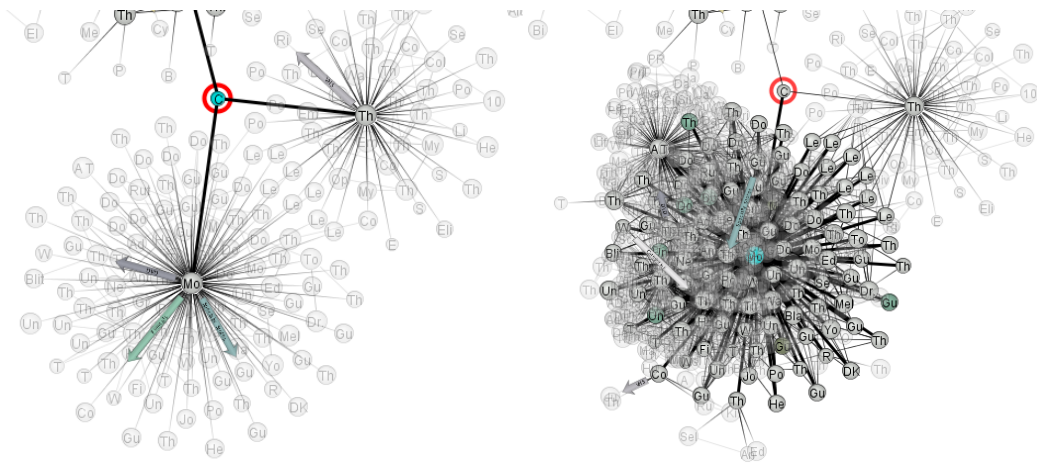


Figure 3.17: Again, a dense star node is being explored. Selecting the node triggers the insertion of a large amount of new nodes that are placed in an already dense area. This does not lead to visual clutter as the new nodes are invisible, but their repulsion forces act on the visible nodes.

3.6.4. Discussion

In this section we describe contributions that increase the quality of incremental layout algorithms. A set of nodes that will possibly become visible in the foreseeable future is already integrated in the layout computation to keep the visual representation stable when they are shown. After they have been placed close to their neighbors, their influence on the layout is gradually increased from almost none to full effect. This keeps the transition between different focal node sets smooth.

The look-ahead strategy we propose significantly improves the layout. Star nodes that are visible have many invisible neighbors that pull the node outwards. This indicates that the vertex is highly connected and possibly worth exploring, even if the connection cannot be seen. On the downside, a high edge to node ratio introduces a large number of nodes on the layout which can cause slowdowns.

Currently, navigating from one part to another and back does not necessarily produce the same visual structure. Whenever a part of the graph is removed from the visible context, the positions of the nodes become invalid. When the nodes are then again added to the context they are attached in a best-fit manner to the existing layout. Reusing the old positions would not help, because the position of the current visual context is different. Section 3.7 describes a layout algorithm that produces such deterministic structures.

3.7. Deterministic Shape Matching

Dynamic graph layouts are often used to position nodes in local views of large graphs. These layouts can be optimized to minimize changes when navigating to other parts of the graph. Dynamic graph layout techniques do not, however, guarantee that a local layout is recognizable when the user visits the same area twice. In this section we present a method to create stable and deterministic layouts of dynamic views of large graphs (Contribution C_3). It is based on a well-known panorama-stitching algorithm from the image processing domain. Given a set of overlapping photographs it creates a larger panorama that combines the original images. In analogy to that our algorithm stitches pre-computed layouts of subgraphs to form a larger, single layout. This deterministic approach makes structures and node locations persistent which creates identical visual representations of the graph. This enables the user to recognize previously encountered parts and to decide whether a certain part of a data set has already been explored before or not.

3.7.1. Background and Overview

According to North [Nor96], layout stability is achieved by minimizing layout changes between consecutive frames. Our definition for layout stability, however, does not apply to consecutive frames alone. In addition we require that the layout of any given subgraph will be the same upon revisitation. Hence we can only claim to solve the second problem here; the dynamic view of a static graph. As a major contribution of this subsection, we specifically deal with requirements 1 and 5. Our solution is a resolution of this conflict fulfilling both requirements.

An important aspect in explorative analysis is about creating and preserving a mental map of the displayed data. We are convinced that the ability to recognize areas that have been visited before

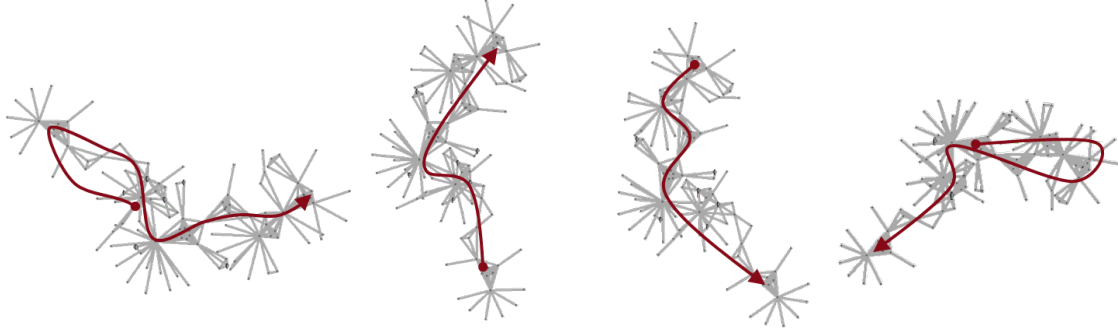


Figure 3.19: *Different exploration paths of the network yield the same visual representation of the node-link diagram. The red spline indicates the path the user took to explore the graph.*

is an important factor in this process. However, conventional force-directed algorithms do not create identical layouts for the same set of nodes from different starting points. On the contrary, they are very sensitive even to small changes in the initial values. As a result, the computed layout of a sub-graph never looks the same. This makes it rather difficult for the user to decide whether an area has already been explored or not, because the visual shape of the topological structure appears to be different.

We propose a dynamic layout that adapts to the currently visible subgraph, but which is independent of the exploration path. The main challenge is to keep the layouts “stable” during exploration. We use a two-level-layout strategy to solve this problem. The first-level layout is done before interactive exploration: We compute a set of overlapping subgraphs which covers the entire graph. The overlapping cover guarantees that most of the nodes will appear in at least two subgraphs. For every subgraph a layout is computed independently to produce the first-level layout. These layouts serve as “building-blocks” for the second level layout that is used during exploration.

The challenge of the second-level layout is to combine these layouts depending on a given visible area of the graph. The node coordinates from different subgraph layouts need to be merged in a deterministic fashion. To achieve this goal we choose a technique that originally comes from the field of image processing: *Panorama stitching* is used to merge a set of overlapping photographs into a single, seamless image. We transfer this technique to graphs. Depending only on the currently visible frame, individual subgraph layouts are selected, weighted and merged to produce the final, visible layout.

To the best of our knowledge, these conflicting requirements have not been solved with a single technique before. Our contribution is a technical solution serving as a proof-of-concept which fulfills both requirements. It extends the notion of layout stability from “frame-to-frame-coherence” to “frame consistency”. This means that the layout of any subgraph looks the same or at least similar for every visit. As a concept, it makes use of existing approaches to create subgraphs and their layouts, but it is not bound to specific approaches. In fact, we believe that this concept offers a design space, which is worth to be explored further in future.

3.7.2. Concept

In this section we will describe how we derive a deterministic global layout from a set of local layouts. We therefore transfer the panorama stitching algorithm to the graph layout domain. Before we can perform our layout stitching algorithm, a set of subgraphs with overlapping node sets needs to be created. A local layout is then computed for every subgraph - independent from the rest of the graph. We refer to these subgraph layouts as *patches* from now on. We then align these patches to match the positions of all nodes that exist in more than one subgraph as good as possible. The node positions that exist in multiple patches are then merged and a unified layout is created. The basic idea is illustrated in Figure 3.20. In the final step, we will explain how to create a layout that consists of more than just two patches.

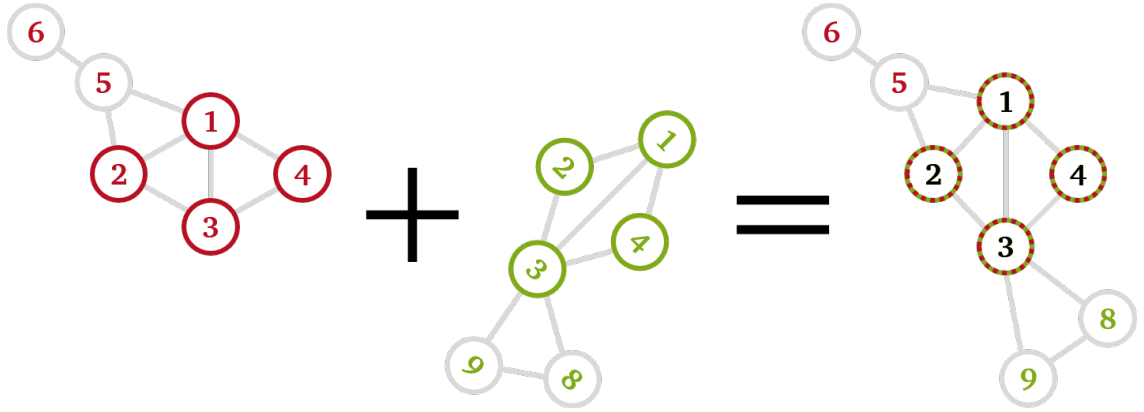


Figure 3.20: The green layout patch is aligned to the red patch using the four shared nodes. Nodes that exist in both layouts are merged, creating a unified layout of both patches.

3.7.2.1. Pre-processing

The series of visible frames is defined by the user who is browsing this graph on a local level. We do not define the means of interaction here, but we assume that the set of visible nodes can be derived from the user's interaction. Our concept defines a local layout for this subgraph. Given a set of visible nodes, the set of layout patches that need to be merged can be derived. The frame which was displayed during the last timestep does not influence the layout of the current frame. This ensures that the same picture is created – independent from the exploration path.

If no set of patches for a graph is provided, we compute a cover of overlapping subgraphs from a topology-based clustering, so that every vertex of the graph is contained in at least one patch. Our approach works with basically any clustering algorithm. However, we point out that the cluster size and content have an influence on their layout which in turn influences the cluster shape that is used for the stitching.

The resulting clusters cannot be used directly as patches, because the clustering typically creates a partition of the graph, i.e. every node is contained in exactly one single cluster only. A straight-

forward approach to make them overlap is to include neighbors of the first degree. In other words: nodes from other patches that are directly connected are added to the node set of the patch. Larger sets are created by adding neighbors of neighbors and so on.

We consider two clusters to be connected if they share at least three nodes. This is the minimum number of points that is required for the layout patch alignment computation.

As soon as the subgraphs are created, a layout is computed for each of them. This can be performed completely independently which allows for using different layout algorithms. Moreover, the computation can be done in a pre-processing step, but also deferred until the layout is actually needed which avoids unnecessary computational overhead. Force-directed algorithms such as that of Fruchterman and Reingold appear to be a sound choice as they reveal local structure and are flexible to integrate user-specified requirements [Kob12].

3.7.2.2. Shape Matching

The sub-layouts are computed independently, therefore the position of nodes is given in a local coordinate system. Nodes that exist in more than one layout generally have different positions in each of them. We will now describe how two patches with overlapping node sets can be aligned so that the distance in between is minimized. Individual node positions do not fit perfectly, but this will be fixed in a later step.

The idea of stitching shapes is based on the work of Brown and Lowe [BL07] who describe an approach for automatic panorama stitching. The authors compute a matching transformation for images based on distinct, but overlapping point clouds. This process is far less complex for graph layouts as no image post-processing such as brightness compensation is required. Most importantly, the point correspondences in the two point sets are known in our setting which simplifies the algorithm.

The second, important contribution comes from the shape-matching algorithm of Müller et al. which works with identical point clouds but in a very different context [MHTG05]. The authors present a method that allows for elastic deformation of three-dimensional objects. With the help of shape matching, the points of the deformed object can be gradually transformed back to their original position. For that, the two geometric point sets are compared and a transformation that reduces the pair-wise distance between all points to a minimum is deduced. Apart from translation and scaling, their transformation scheme offers refinements such as twisting and compression which are not present in the work of Brown and Lowe.

We trivially acquire a set of vertices that exist in two given layouts by computing the intersection of the two sets.

As long as the set contains at least three vertices, we can use the standard least-squares fitting method [AHB87] to compute a deterministic matching transformation. For the sake of simplicity, we will restrict this computation to rigid transformation, i.e. rotation and translation, but general affine transformations are feasible as well. For every point p in sublayout A we specify its counterpart p' in sublayout B as

$$p'_i = Rp_i + t + \epsilon_i \quad (3.11)$$

Here, R is a rotation matrix, t a translation vector and ϵ the measure of error. After solving for ϵ and accumulating the error over all points, we get to the following equation:

$$\epsilon^2 = \sum_{i=1}^n \|\epsilon_i\|^2 = \sum_{i=1}^n \|p'_i - (Rp_i + t)\|^2 \quad (3.12)$$

The error becomes minimal if both point clouds have the same centroid [AHB87]. This can be achieved by subtracting the centroid of their respective sets (denoted as c_p and $c_{p'}$) from the point locations. The task is now to find an optimal rotation matrix where the pair-wise distance is minimal for all points.



Figure 3.21: Rotating one of the two point clouds (green) reduces the average distance between pairs.

This matrix can be deduced from a 2×2 cost matrix H that measures the distance between two point clouds. We subtract the centroids from both data sets to bring them to the origin and define this matrix H as

$$H = \sum_i^n (p_i - c_p)(p'_i - c_{p'})^T \quad (3.13)$$

Using singular value decomposition (SVD), the matrix H can be factorized into two rotations U and V and a diagonal scaling matrix S .

$$[U, S, V] = \text{SVD}(H) \quad (3.14)$$

See, for example, the introduction by Wall et al. [WRR03] for details on the mathematical background of the singular value decomposition. The final desired rotation matrix can be computed as:

$$R = VU^T \quad (3.15)$$

The final result is a transformed point \hat{p}_i that represents the point p_i of sublayout A in the coordinate system of sublayout B .

$$\hat{p}_i = R(p_i - c_p) + c_{p'} \quad (3.16)$$

The accumulated difference between \hat{p}_i and the original point p'_i relates to the previously computed error ε and can be used as a quality measure for this transformation process.

3.7.2.3. Combining Multiple Shape-Matching Transformations

The approach we presented so far works well for combining two patches. However, in general, a frame consists of more than that. We therefore describe how to stitch multiple patches in one frame and how we create a smooth transition between two consecutive frames. The problem we solve here is to find a deterministic order in which the patches are stitched. Therefore, we create a meta-graph of the patches. Two patches are connected, if two patches share common nodes (see Figure 3.23 left and center). Thus, they can be stitched together. For the remaining part of this thesis, this graph is referred to as *patch graph*.

If the patches that should be merged are connected directly, only a deterministic order of stitching operations needs to be defined. Otherwise, also a connecting series of patch stitchings must be generated to ensure that also distant patches can be combined.

This series of stitchings of overlapping patches can be seen as a path in the patch graph. Also, on this level of abstraction, the interactive exploration can be seen as a user-driven traversal of this patch graph.

Starting with a single patch, the user continues exploring, eventually reaching a part of the graph, that cannot be visualized without including additional patches in the visible subgraph. Adjacent patches are then added until the requested graph region can be visualized. This graph traversal must be stateless and therefore independent of previously visible patches. If this was not the case, different exploration paths would have different stitching orders thus result in different global layouts. Three possible setups are depicted in Figure 3.22.

The reason for this is that the patch graph contains multiple paths that connect the visible patches. Reducing the number of edges naturally leads to a reduction of the number of paths. To enforce a stable matching order, we remove all edges from the patch graph that are not strictly necessary to keep the graph connected (see the illustrations in Figure 3.23). What is left is a spanning tree of the graph and can be computed by Kruskal's algorithm.

It is also able to incorporate edge weights, thus computing the spanning tree with lowest total weight – the minimum spanning tree (MST). Starting with a graph that contains all nodes but no edges, edges with the lowest weight are continuously added as long as they don't lead to cycles in the graph.

Although the error value of each matching seems like a natural choice to maximize the quality of the whole layout, several drawbacks lead us to the decision against using it. First and foremost, using the matching error as edge weight is possible only if the matching error was known for all pairs of connected patches. Computing the optimal affine transformation of all possible combinations of layout patches is rather time-consuming. Furthermore, interactive manipulation of a single patch layout would result in changing weights for its incident edges, which in turn could cause changes in the spanning tree of the patch graph.

Instead, we define a similarity-based weight function so that edges between pairs of patches with large overlap ratios have lower edges weights. They are then most likely to be stitched first. The Jaccard coefficient is a measure that indicates how similar two sets are and is defined as:

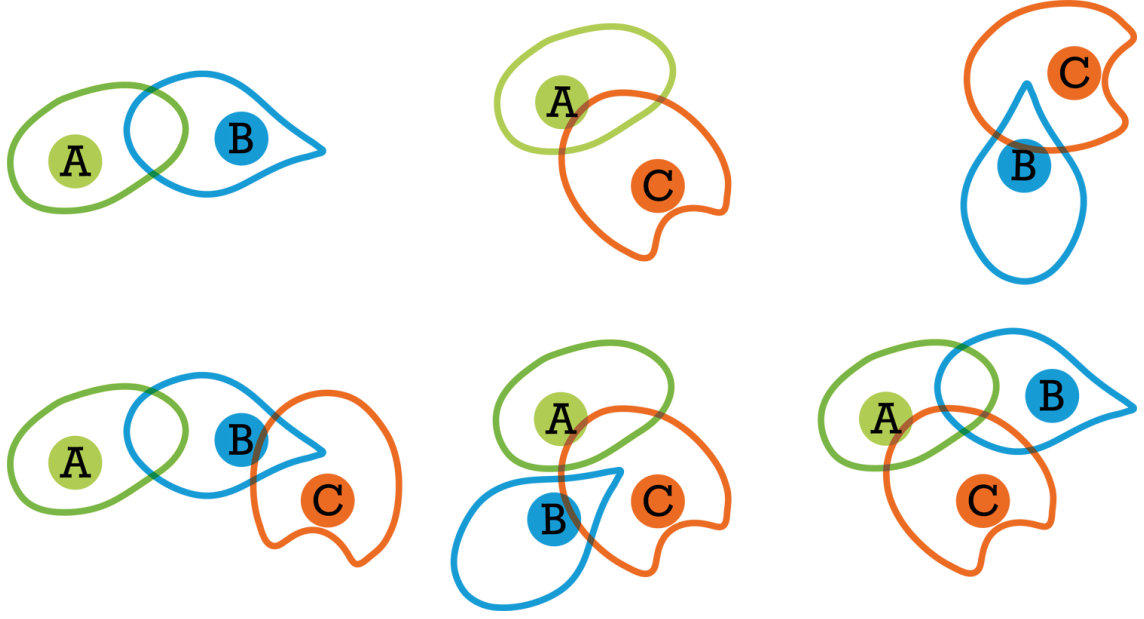


Figure 3.22: Three patches (A, B, C) with corresponding 1-to-1 stichtings (top row). If the patches would be stitched in the order they become visible, different stitched layouts would result. In this configuration three different global layouts could be produced (bottom row).

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (3.17)$$

This measure does not require the computation of the matching error between all edges in the patch graph. Similar to the distance function that is often used in graph clustering, we can use this (or another) similarity measure and assign this value to the edges of the patch graph.

For every pair of layout patches in this spanning tree, only one single path exists. These paths in the tree are no longer the shortest in general when compared to the original patch graph, but this reduction in freedom results in consistent patch stitching chains. This also ensures independence of previous frames, as the stitching order is fixed. We use the root of the spanning tree as end point for all paths. Thus, every visible patch is stitched to its parent patch until the root node is encountered. This is a critical aspect as it ensures that patches are always matched to the same neighbor patches. The local position of a node is thus transformed by the series of affine transformations of the patches along the path to the root patch.

Some nodes belong to multiple patches and would, without additional correction, have multiple positions on the drawing canvas. We therefore derive from all these positions a commonly shared, unique position. In such cases, we use a linear combination of the nodes' weight factor to place the nodes depending on time and the user focus. This ensures a smooth transition from one layout frame to another.

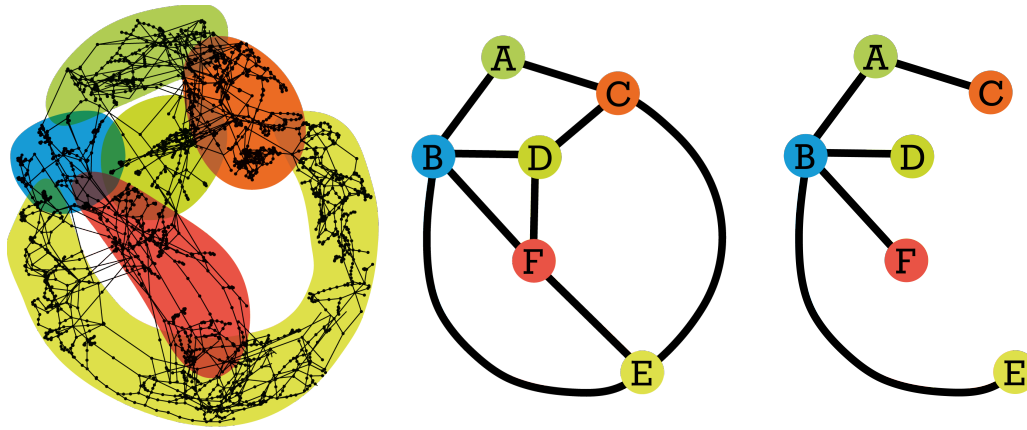


Figure 3.23: *The original graph is reduced to a graph of patches which is then reduced to a spanning tree. This tree is used to define unique paths between any two nodes.*

3.7.3. Preliminary Tests

In this section we will present some test results of both artificial and real data sets. First we demonstrate the concept in detail using a basic test graph. Second, we use a real data set to demonstrate that different exploration paths result in congruent layouts.

3.7.3.1. Concept Verification

The first test run is based on a graph of the form of a Venn diagram for three sets (see Figure 3.24). It contains three node rings that overlap at the center. This graph is small yet complex enough to test the correctness of our approach. Its structure allows, on the one hand, the extraction of three overlapping patches – the rings – and ensures, on the other hand that their layouts overlap only very little while having excellent matching error scores.

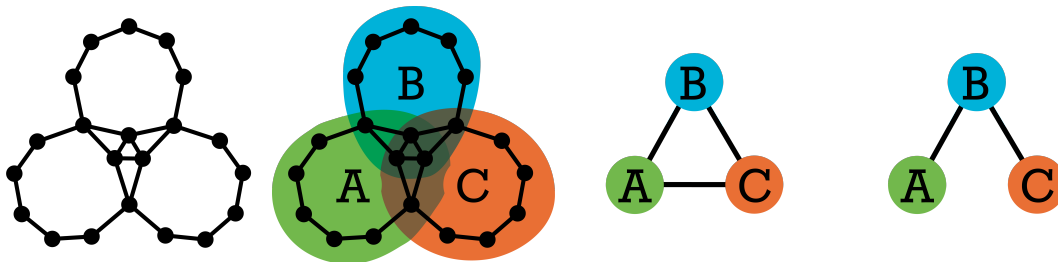


Figure 3.24: *From left to right: The Venn diagram (1) is split into three overlapping subgraphs (2). These form a patch graph (3) with 3 patches and 3 edges which is reduced to a tree (4) to enable a stable interactive exploration.*

We create the patch graph and compute a spanning tree. Using the tree, we then merge one patch after another in coordination with the interactive exploration component. The green patch is shown first and thus forms the root patch for rendering and remains as it is (Figure 3.25 left). The blue patch is flipped, rotated and translated to the bottom of the green, minimizing the matching error between the green and the blue patch. The common nodes are then merged, creating the layout in Figure 3.26 center). In the next step, the orange patch is aligned with the blue, already aligned patch. This results in a total transformation (Figure 3.26 right) of about 180° for the orange patch. Lastly, the node positions are unified where necessary and merged for the final, visible graph.

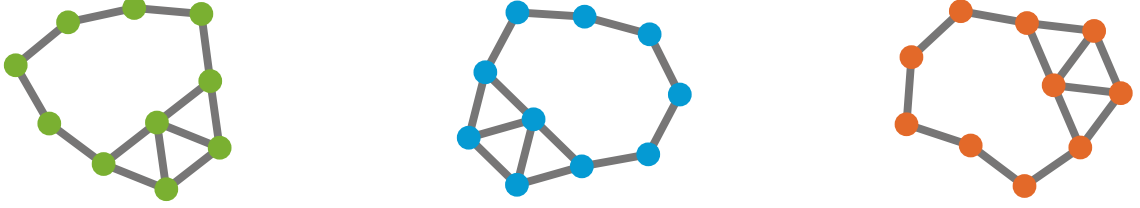


Figure 3.25: Individual layouts of the three patches of the Venn diagram graph.

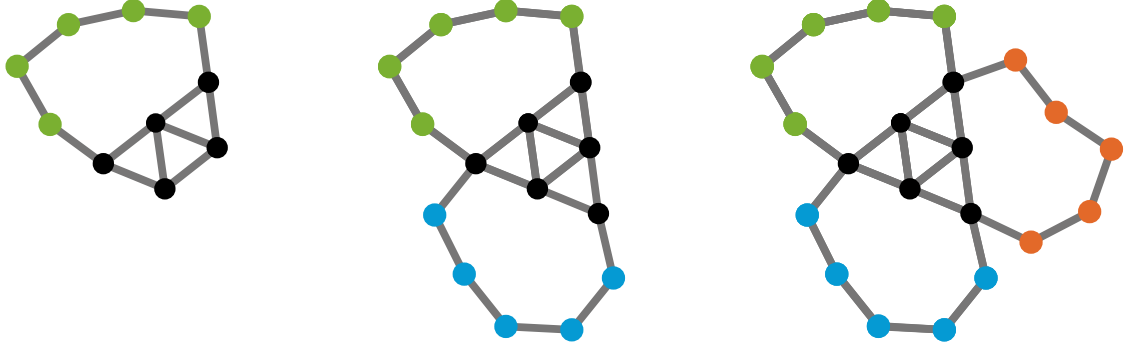


Figure 3.26: Initially, only the green layout is visible (left). In the next step, the blue patch is matched against the green patch (center). Finally, the orange patch is matched to the blue patch (right).

The second test we performed was with a pair of star-shaped subgraphs which has been extracted from a real data set. The layout of both subgraphs is strongly affected by the high degree of the central nodes. The intersection of the node sets contains only four elements, but both star nodes are included. This leads to a significant overlap of the patches, but the star patterns are still visible (Figure 3.27).

In a similar data set, two star-shaped graphs have been extracted again, but this time, they intersect only at their boundaries.

As can be seen in Figure 3.28, the patterns are stitched with only very little deformation of the original shapes. More importantly, the nodes that form the connection between the two clusters are clearly distinguishable in the stitched layout.

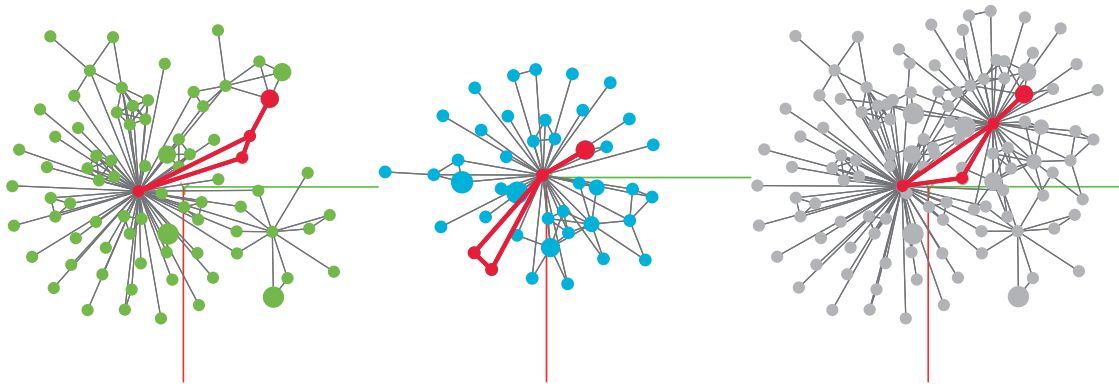


Figure 3.27: Two star-shaped patches (green, blue) are stitched together forming a single graph layout (gray). The common nodes (red) that form the base for the matching are the only nodes that are distorted. Although the central nodes are in both sets, both patches are recognizable in the stitched layout.

3.7.3.2. Exploration Independence

The claim of this contribution is to create a deterministic layout which is independent of the exploration path. We test this hypothesis by navigating through several clusters of a larger graph in different order and compare the generated layouts.

The data set for this test is a network graph from the medical domain [GCV*07] with roughly 1.5k nodes, 5.5k edges. Our clustering algorithm created 77 layout patches. The size of the graph features a fair amount of complexity while still being visually comprehensible when viewed as a whole (Figure 3.29). We used a force-directed layout of the whole data set to display the exploration path as ground truth and compare the results. With only one single parameter – the number of iterations – the *chinese whispers* algorithm [Bie06] appeared to be a good choice for the clustering of this graph.

Several different explorations have been performed to verify the validity of our approach. They all started at different points exploring the same clusters, but in different orders. As can be seen in the teaser figure on the first page, the resulting stitched layouts are congruent. The construction of two exemplary stitched layouts is depicted in Figure 3.30 and Figure 3.31, respectively.

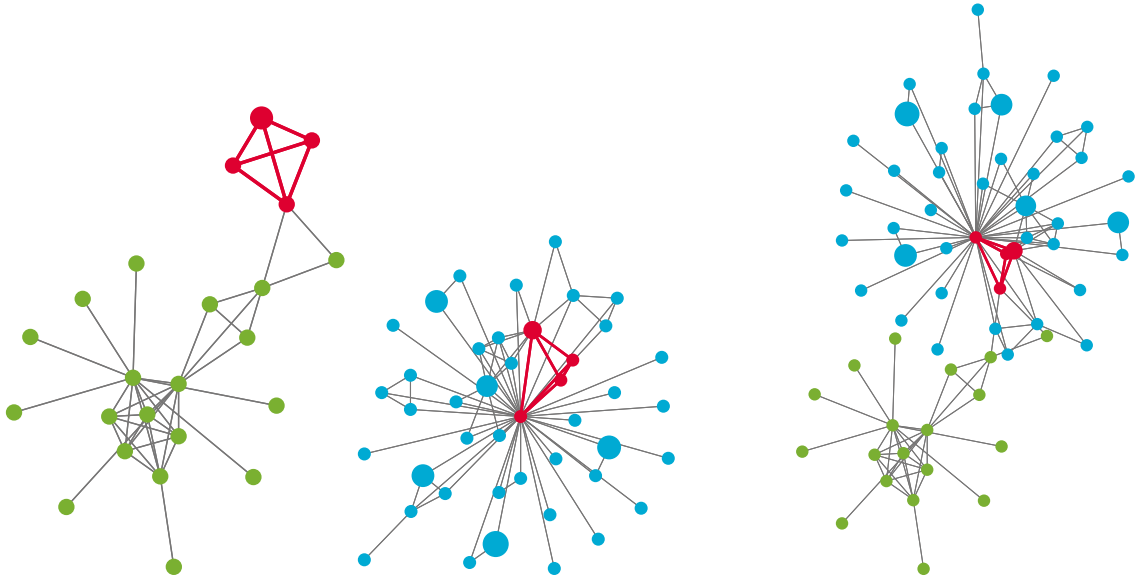


Figure 3.28: Two star-shaped patches (green, blue). A clear outlier region in the green patch leads to a clear separation in the stitched layout (right).

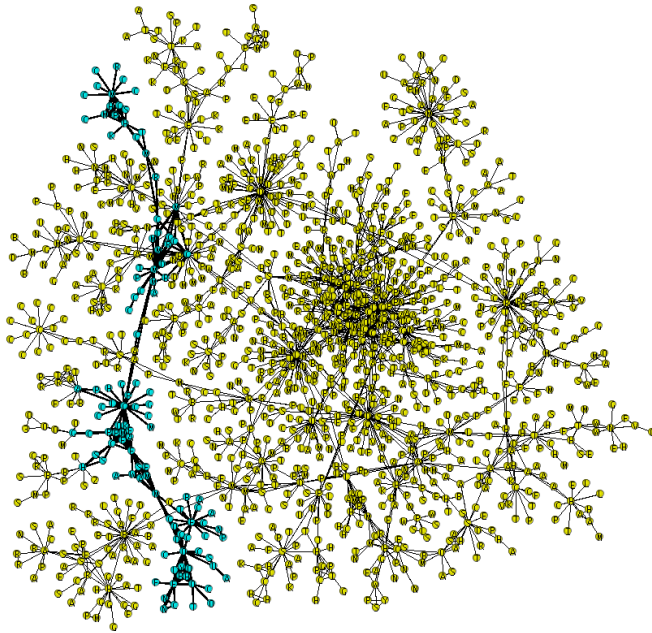


Figure 3.29: The entire graph laid out with a force-directed placement algorithm. The exploration path in this global layout is highlighted in cyan and appears to be quite different from the stitched local layouts.

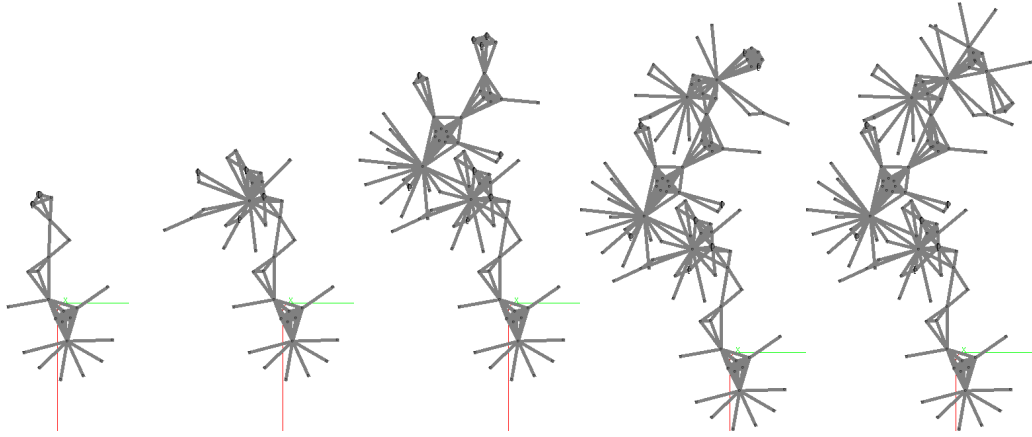


Figure 3.30: *The first exploration through five clusters in the order 1, 2, 3, 4, 5.*

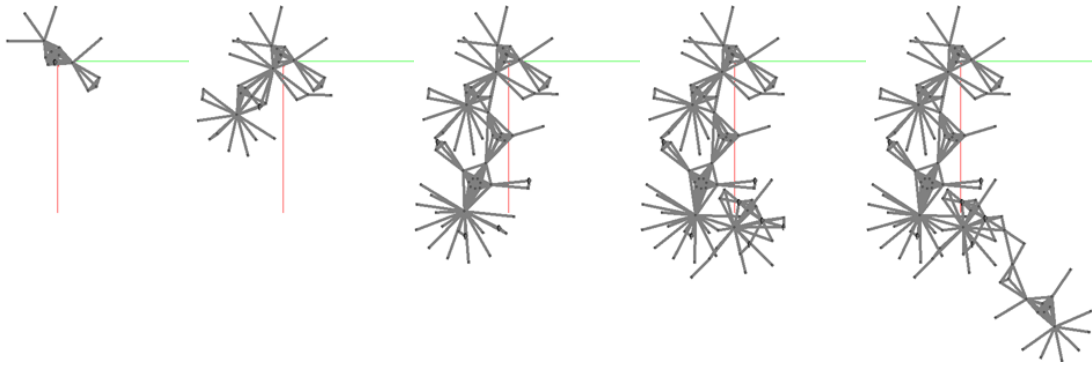


Figure 3.31: *In the second run, the clusters were explored in the inverse order resulting in a congruent layout.*

3.7.4. Discussion

In this section we presented a new approach that aims to create dynamic graph layouts which are independent of the exploration path. It works independent of specific layout algorithms and thus also works for highly dynamic force-directed layout algorithms. When the user explores large graphs with dynamic views, new nodes are typically added in the proximity of existing, linked nodes.

This approach is thus highly dependent on the exploration path – the layout can look very different even for very similar explorations. Our method overcomes this limitation with techniques from the computer vision domain where image stitching is used to merge multiple photographs with overlapping areas into a larger image. In analogy to that, our method uses pre-computed layout patches that are sewn together in deterministic order. Consequently, the resulting layout is stable, independent of the user’s exploration path and will, thus, always look the same. In contrast

to many other dynamic graph layout algorithms, a fair amount of computational effort can be pre-computed which increases the interactivity and reduces the workload at runtime. Being able to work with different layout algorithms for different patches makes it also very versatile.

Compared to conventional layout methods, the additional computational effort is also rather small. The cost of layout computation is increased by the factor of nodes that exist in multiple matches. Runtime costs are limited to the creation a 2×2 cost matrix and its decomposition which has a constant running time [MHTG05].

The layout stitching method we presented sees the subgraph as a disconnected point cloud and merges the patches without respect to the topological structure. Closely related to that, it also ignores the points that are not in the intersection of the two nodes sets. As a result, two layouts could be aligned so that the disjoint parts overlap as well which is undesired.

We assume that more sophisticated approaches for the computation of the patch overlaps could mitigate this problem and improve the stitching quality. This includes the use of the graph topology metrics such as connectivity to find and include the best-fitting nodes. An ideal strategy would include nodes that emphasize certain visual features of the cluster layout to make the structure memorable.

3.8. Deterministic High-dimensional Layout Stitching

In contrast to the state-of-the-art approach that operates in the 2D screen space only (see Section 3.7), we perform this process in high-dimensional space before projecting the results into the 2D plane (Contribution C₃). This gives additional degrees of freedom and consequently a smoother transition process between two consecutive frames. It is rather difficult for the user to decide whether an area in a conventional dynamic graph view has already been explored or not. The visual shape of the topological structure is different even though the structure itself is the same. This problem has been addressed already in Section 3.7, based on a divide-and-conquer approach. However, one major shortcoming of this original approach is that the merging operation is not very stable. The positions of the nodes that are used for the merging operation in one patch are often very different from those in the other subgraph. Due to the fact that only a two-dimensional affine transformation is used to bring them together, large divergences remain.

The claim of this section is to improve the original work so that stable stitchings can be created that were unstable before. It goes beyond the original work by solving the stitching problem in high-dimensional space. The process involves the layout computation and the merging of two layout patches in order to get the aforementioned additional degrees of freedom. This stabilizes the stitching operation but also requires finding an appropriate measure to project the layout back into the 2D plane. We interpolate the points of two consecutive frames linearly to create a smooth transition between two exploration steps.

3.8.1. Concept

In this section we briefly outline the idea of 2D layout stitching and explain how we transfer this idea into high-dimensional space and project it back into the 2D plane (see Figure 3.32).



Figure 3.32: The original graph is first partitioned into clusters which are then used to create overlapping patches. For each pair of overlapping patches a linear transformation is computed that brings shared nodes together. They are then merged before they are projected into screen space.

As already described in Section 3.7, we first cluster the graph data set into smaller, overlapping patches. Adding directly connected nodes from the border of a patch creates overlaps that are required for the stitching process (see Figure 3.33). Two clusters are considered as being connected, if and only if at least one edge exists between the two. Up to this point, we follow the original version of the approach.

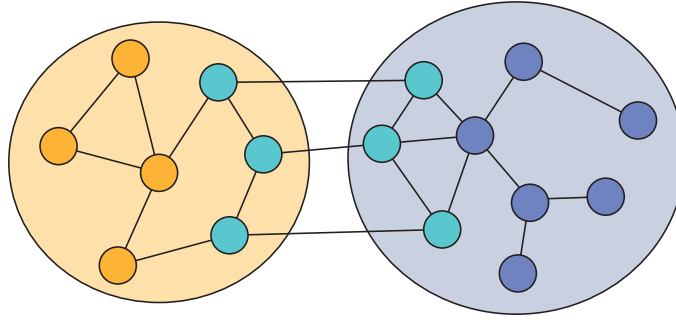


Figure 3.33: Nodes that are directly connected to nodes from other patches are used to create overlapping patches.

In the next step, an individual layout is computed for every patch in d dimensions. We use a straight-forward extension of the two-dimensional Fruchterman-Reingold algorithm to compute a high-dimensional position vector for every node. However, other methods such as that of Gajer et al. [GGK01] should work at least equally well. The algorithm runtime does not play a major role here, as this can be precomputed.

3.8.1.1. Layout Stitching

Layout stitching defines the process of merging two overlapping layout sets. First, an optimal linear transformation between the node positions of overlapping patches is computed. We prefer linear over affine transformation as it preserves the shape better. We denote d as the number of dimensions that is used for the graph layout. To define a linear transformation between two patches in d dimensions, at least $d + 1$ nodes which are shared among the patches are required. In that case, we consider these patches as overlapping. For two dimensions, three points are sufficient to derive a unique linear transformation. It brings the nodes that exist in both layout but with different position coordinates as close together as possible. If the patches share more

than $d + 1$ nodes, we calculate the linear transformation which is optimal with respect to the given constraints.

This transformation is computed as follows: based on a least-square fitting method we define a target function that we want to minimize. This function measures the distances between the two corresponding positions of a node. We normalize both data sets by first moving the centroids of the two patches to the origin. As a side product, this already gives the translation vector of the transformation matrix. Then, a $d \times d$ cost matrix is created based on pair-wise distances. Variable p_i and p'_i denote the positions of node i , c_p and $c_{p'}$ represent the center of gravity of their respective patches.

$$H = \sum_i^n (p_i - c_p)(p'_i - c_{p'})^T \quad (3.18)$$

This matrix is then factorized by Singular Value Decomposition (SVD) so that the rotation component can be extracted. Applying the SVD splits the matrix into two rotation matrices U and V and a diagonal matrix. The rotation part R of the transformation is defined as VU^T . The original work of Steiger et al. contains a more detailed explanation of this computation.

As soon as the transformation matrix has been computed, two overlapping patches are stitched together by merging these duplicates into single positions. This is required, because nodes that exist in both layouts have a position in each of them and the linear transformation does not bring them to the exact same spot in general. We therefore use a linear interpolation of their weighted influence on the current view to derive a smooth transition between old and new node position.

3.8.1.2. Projection

In contrast to the original work, the node positions are still in high-dimensional space. Therefore, the dimensionality of the stitched point cloud must be reduced to the two most-relevant dimensions before it can be displayed on the screen.

We use non-linear multidimensional scaling (MDS) to achieve an optimal projection while preserving distances [Tor52]. It uses a normalized stress function to define the difference in distance between the original and the projected space. This function is then minimized using optimization algorithms. From our practical experiments, the stress-based MDS (also referred to as SMACOF) achieves high-quality results. However, depending on the data set, other projection algorithms may perform better. A comparative overview of different dimensionality reduction methods can be found in the work of Joia et al [JPC*11]. Since this step of our approach is transparent, it does not depend on any particular projection algorithm.

This projection is defined only up to rotation and scale, so additional constraints need to be added to keep these two constant across multiple operations. These free variables can be fixated by using the screen's display area. Based on the 2D screen coordinates, a principal component analysis (PCA) gives the two axes with the largest variances. Using rotation and scaling, they are aligned so that the larger goes from left to right, the smaller from bottom to top. This order is chosen to benefit from the fact that typical screen displays' width is larger than their height.

Nonetheless, the computed projection matrices for the old and new point cloud can be quite different. This causes strong sudden changes in the picture which makes it difficult to follow the transition process. We therefore interpolate the points of two consecutive frames linearly.

In this section, we outlined how we perform layout stitching based on only two overlapping patches. Creating a combined layout consisting of more than just two patches is not as straightforward as it may appear, but we follow precisely the strategy of [SLTM*13] here.

3.8.2. Proof of Concept

We conducted some tests to demonstrate the validity of our approach. For the purpose of the illustration, the individual layouts were projected into 2D. In the actual implementation, only the final, merged layout is projected. We performed the projection using classical MDS and a stress-based majorization algorithm. Projecting the points using stress majorization (SMACOF) methods is computationally more expensive, but yields a better spread between the nodes and thus a cleaner picture. This is why only the results of this approach are discussed. The first experiment was performed using an artificial graph (see Figure 3.34).

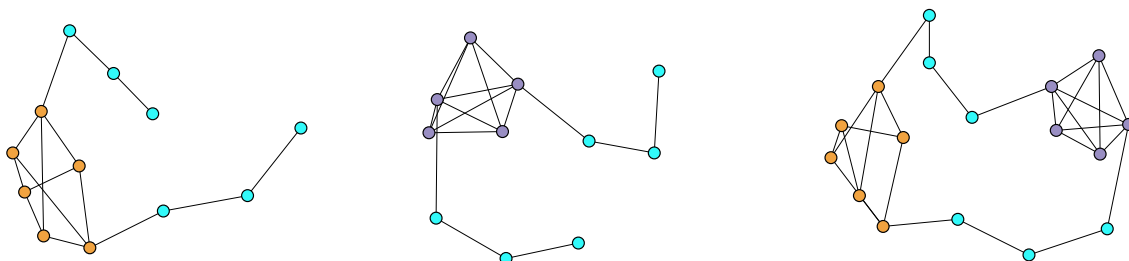


Figure 3.34: *The subgraph on the left side contains six unique nodes, the subgraph in the center contains a five-clique. Six nodes (in cyan) are in both layouts. The merged and projected results are depicted on the right side.*

The other tests were performed based on the Diseasesome² network data set. The second test was based on two patches with significant overlap. Most nodes were part of both layouts, which lead to highly similar shapes. As can be seen in Figure 3.35, the projection of the merged layout preserved the similarity in the new, integrated layout.

The third test graph comprises two patches with 152 and 104 nodes, respectively. The patches have only nine nodes in common. Two independent layouts were computed in 5D layout space, then stitched together at the shared nodes and projected. The result can be seen in Figure 3.36. A large part of both patches remain clearly separable, while the overlapping areas appear cluttered in the 2D view. However, the visual representation is deterministic, because the layouts of the individual patches are fixed at runtime. Also, the problem that the transformation flips one patch which occurs frequently in the 2D version does not appear here due to the larger number of degrees of freedom.

²<http://diseasome.eu/>

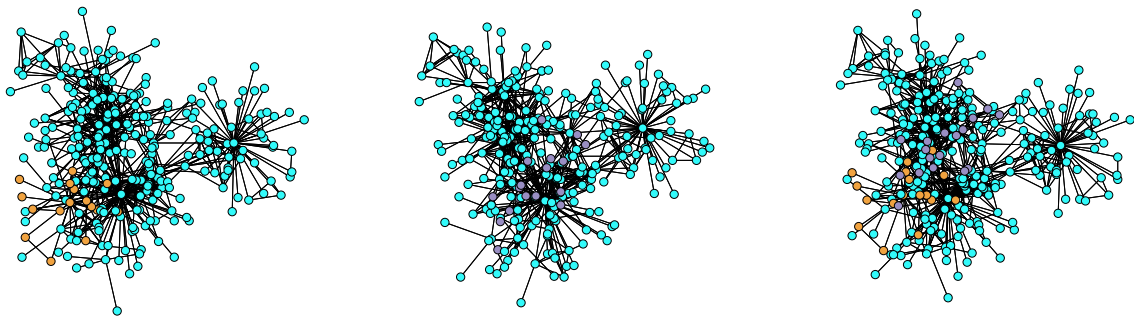


Figure 3.35: Two subsets of the Diseasesome network which share more than 90% of the nodes. The merged layout on the right integrates the purple nodes (center image) into the existing layout, preserving the general shape of the existing layout (depicted on the left).

3.8.3. Discussion

In this section we introduced an extension of the 2D layout stitching approach that works in high-dimensional space. This brings additional freedom for the matching of precomputed layout patches. We compute the stitching and the merge of duplicate position vectors in high-dimensional space before we project the resulting point vectors back into screen space. Using linear interpolation ensures a smooth transition between node positions before and after the stitching process. This enables the user to trace the changes in the layout as in the 2D approach, but the algorithm in the background is more stable.

3.9. Summary

In this chapter we discussed *Local Graph Views*, a group of techniques for visual graph analysis that is based on the idea that local features are most relevant for the data scientist.

One of the major drivers is the massive increase of data volumes, but also increasing data inter-connectivity makes analysis more complex than ever. Keeping an overview, extracting the most relevant aspects and gaining insight are prominent requirements that today's decision-support tools must fulfill. These high-level expressions are split into a discrete set of explicit requirements in Section 3.2. Of course, these problems and requirements have existed for a while and have been tackled previously. In Section 3.3, previous approaches are discussed, grouped by category. Most of them use node-link diagrams to depict network data and provide interaction support to manipulate data or representation. An overarching goal is to gain insight in large data sets based on intuitive, graphical displays.

Based on the corpora of scientific contributions, several gaps have been identified. In Section 3.4 basic definitions of graphs are given and some mathematical background is laid out to define a common baseline for the chapters that follow. Sections 3.5 – 3.8 specifically deal with these shortcomings and provide approaches to fill these gaps. In each of the sections, a detailed explanation of the contribution, followed by a discussion is given. We will now review this chapter with an analysis of whether the requirements that were defined in Section 3.2 are fulfilled.

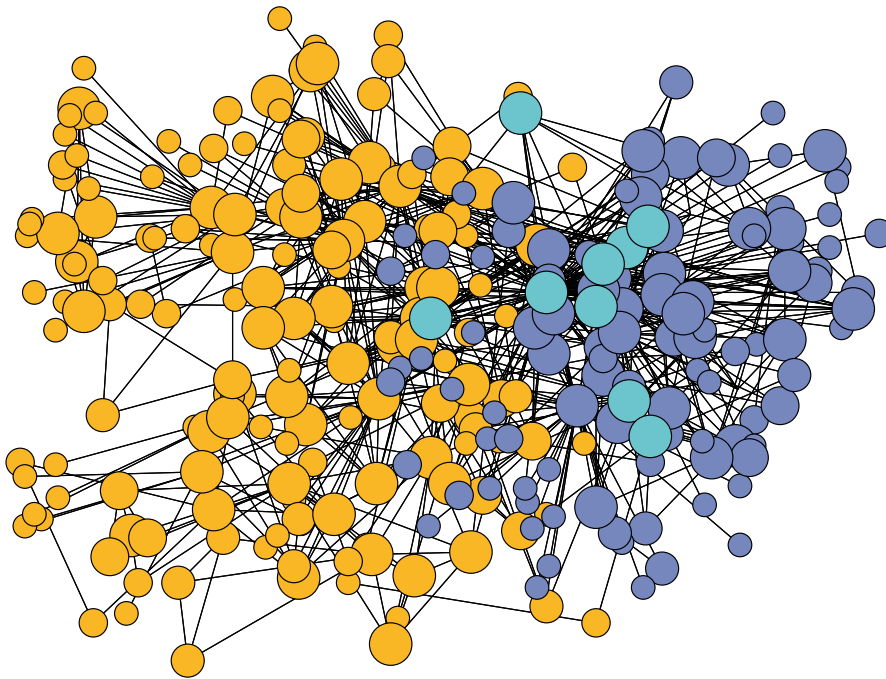


Figure 3.36: *Two overlapping patches, stitched in 5D space and projected with MDS. While the structures are still separable, some visual clutters appear in the area of the intersection. Using a smaller number of points for the stitching process and adding an offset to one of the patches mitigate the problem.*

- **Requirement 1 (Complexity Reduction):** This requirement has been fulfilled by using local views of a graph rather than showing the entire graph in one picture. This forms the basis for all approaches that were presented in this chapter.
- **Requirement 2 (Orientation):** We support the user's orientation with signposts glyphs that indicate directions in which relevant or interesting parts of the graph lie with respect to the current layout algorithm. Details can be found in Section 3.5.
- **Requirement 3 (Navigation):** The user can navigate through the node-link diagram and explore other regions of the graph with the help of the signposts (Section 3.5)
- **Requirement 4 (Smooth Transitions):** Using a fore-sighted layout with ghost nodes, spring forces are minimized in the layout. See Section 3.6.
- **Requirement 5 (Determinism / Familiarity):** The signposts approach is complemented by a second approach that reconstructs previously explored parts of the graph by stitching pre-computed layout patches (see Sections 3.7 and 3.8).

We can therefore conclude that all requirements for the successful exploration of networks in the most general sense have been fulfilled. With these contributions as a basis, we can turn towards

a more complex, but highly relevant structure for many practical applications: sensor networks. In contrast to general networks, each node is attributed with a time series element. The electric grid is an example of such a network, as it inherently exhibits network structure, but also provides temporal information at each of the transformer stations. This information could be the current load, voltage or the temperature of cables and transformers that changes over time. We will look into this in detail in the following chapter of the thesis.

4. Visual Analysis of Sensor Networks

In this chapter, the focus is narrowed from general networks down to *Sensor Networks* and we will investigate different approaches for the visual analysis of these structures. Sensor networks are a subtype of graphs that contain – aside from the edge structure – temporal data for each of the nodes.

In the frame of the following sections, a visualization environment is constructed that aims to support the analyst of such data to gain insight in the observed data sets. Following the idea of Visual Analytics, which was presented in Chapter 2 and the concepts for graph analysis in Chapter 3, a set of analysis tools will be presented in the following sections. This comprises the integration of automated systems, but also the human user to benefit from the advantages of both worlds. The analyst is enabled to track the machine-based analysis process, identify trends and patterns in the time series domain, find correlations in the network structure and simulate the effect of modifications in the network.

The first section of this chapter motivates the work from a practical perspective and provides some background information. In analogy to Chapter 3, a set of requirements is defined on that basis. They are necessary for the analyst to successfully work with such an analysis environment, gain insight in the critical areas and find appropriate countermeasures. Following this, related work is presented, split into different categories such as fully automated approaches versus visualization-based ones. A dedicated section discusses the importance of network layouts in a mixed geographic and topological context. This is highly relevant for many applications, since sensor networks are often laid out and displayed in a geographic context while trying to preserve the inherent topology.

The following sections explicitly discuss a series of contributions to the scientific domain. Starting with the initial setup of a visualization environment, the integration of an analysis pipeline on the basis of an expert system is discussed. Several interaction metaphors such as brushing and linking and drill-down concepts are combined to provide different perspectives on the data. It also integrates a variety of external data sources to provide a quick overview for the network analyst. In the next iteration, we encode the similarity of time series data in color so that the analyst can identify trends and patterns in the data. A case study on a real-world data set based on load measurements in an electric grid illustrates the usefulness of this analysis system. Two-dimensional color maps play a key role as they map geometric distance to perceived color differences. Finding the ideal color map for different tasks is vital for the success of the analysis of patterns. We therefore present the ColorMap-Explorer, a visual tool for the analysis and comparison of color maps. During the course of the analysis, the domain expert might identify weaknesses in the network. In order to identify the best reinforcement strategy, different alternatives in terms of structural modifications must be evaluated. We present in the last contribution of this thesis a tool that enables the analyst to compare such changes in a side-by-side view and on a per-node basis. This is illustrated

along a use-case in a fresh-water supply network. Finally, and again in analogy to Chapter 3, we reflect the requirements that have been defined in the context of the corresponding contributions.

Parts of this chapter have been published [SMDK13a, SMDK13b, SHS*14, SBM*14, SBMK14, SBM*15].

4.1. Background

The amount of sensor data has seen a rapid growth over the past years in many different applications and scenarios. In this thesis, we focus on univariate sensors that measure a single variable over time. In many applications it can be safely assumed that the sensors measure variables that are explicitly or implicitly linked. We motivate our approach with a practical example: the exploratory analysis of the power consumption in an electrical grid. This involves on the one hand an in-depth analysis of the nature of a given network structure and on the other hand large amounts of sensor data that is recorded at the nodes of the network.

Over the past years, the management of geo-based infrastructures such as gas and water pipeline networks gained traction in both research and application. Of particular interest are electric grids which are currently converging into *Smart Grids*. Not only since some European countries decided to enforce the use of renewable energy there has been a paradigm shift towards a bigger amount of smaller plants distributed over large areas. A remarkable trend towards renewable energies drives the innovation in the electricity supply chain. This trend causes drastic changes in the grid, leading to a rapid increase in the grids' complexity.

Since its inception at the end of the nineteenth century, the electrical grid has continuously grown in size and complexity. The basic idea behind its construction was, that electricity was generated at a few central sites – typically large power plants – and then distributed across the country to its consumers. High-throughput and high-voltage lines at the top carry the energy over large distances to local hubs. From there, a series of medium- and low-voltage grids distribute electricity to individual consumers. The resulting distribution infrastructure is clearly hierarchical.

Unfortunately, this hierarchy is becoming less and less suitable for today's requirements. A paradigm shift away from large, central power-plants in favor of many, small and highly heterogeneous energy producers is in progress. The factors climate change and security are major drivers for the demand for local, renewable energies in Europe, the US and other parts of the world. In strong contrast to conventional electricity generation, the renewable energy providers can only supply small amounts of electricity. The dependence on environmental factors such as solar radiation or wind leads to strong fluctuations over time as well. Naturally, these factors were not taken into account when the electric grid was constructed [IL08].

Also, deregulation caused a continuous growth in the diversification of stakeholders which significantly increases the complexity of grid management as more and more partners are involved. In order to be able to deal with these challenges, novel solutions need to be developed in the engineering domain, but also in grid management and monitoring systems to keep this sensitive system within permissible limits. The German ministry for trade and economics states² that Europe's electric grid infrastructure currently faces three major challenges:

²<http://www.bmwi.de/DE/Themen/Energie/Stromnetze/stromnetze-der-zukunft>

- The generation of electricity is becoming more and more erratic which is noticeable for wind energy plants in particular. This can lead to deterioration in the stability of the grid in both transmission and distribution.
- The direction of flow is no longer necessarily from the plants through the transmission and distribution grids to the consumer. With increasing decentralized generation, the grid has to be able to deal with inverted flow directions.
- The cross-nation trade of electric energy will increase, which, in turn, will increase the management effort, especially for transit countries such as Germany.

Typically, the operators in today's control rooms use *Supervisory Control And Data Acquisition* (SCADA) systems to monitor and control the electric grid whereby a network of sensors measures its status variables. These systems are event-based, involving messages that contain sensor readings and other status update notifications.

At present, large numbers of sensors are installed in the transmission level of the grid hierarchy [CPF09]. An increase in decentralized generation will lead to an increased power input at the medium and low voltage levels. This, in turn, necessitates the installation of sensors at these levels in the hierarchy. The throughput in today's SCADA systems is already high and will increase dramatically as more and more sensors are installed at the lower levels of the grid [KSH07]. With increasing net flows through the grid and limited financial options, it has also become necessary to operate the grid closer and closer to its limits. This lead to an additional increase of the already large number of alarms the operator has to deal with [HAB09]. The increase in information brings great opportunities but requires dedicated analysis support.

Today, the majority of electricity is still produced at a few, large power plants. From there, the energy is then transferred over large distances using high-voltage lines. Low voltage distribution grids carry the electricity over the final mile to the customers. However, things are changing: over the past years, a noticeable trend towards renewable energies has been registered in most industrialized countries. These new source of electric energy comprise wind turbines, biomass plant and solar plants. The majority of installed solar plants are rather small – consequently, the number of stakeholders is rather large and the site locations are distributed across the entire country. Taking into account that their input is directly dependent on the solar radiation and therefore highly fluctuating, new challenges arise not only on the engineering side but also for its management and monitoring.

4.2. Requirements

Network operators monitor the network, trying to prevent failures and detect events, before they cause cascading effects. In case of failure, connections must be checked and alternative routes must be identified. Clear and easy-to-understand maps play an essential role in doing so. Such maps are still often crafted manually and whenever the grid topology changes, the corresponding map must be updated. To date, these changes are often tracked manually. With increasing number and requirements for shorter latency times, this becomes more and more expensive in terms of

time and effort. Updating maps by hand is not only costly, but becomes also increasingly difficult with the number of nodes. This makes response time a factor that is today much more important than it was a few years ago.

In the case of geo-based networks, a natural layout already exists: the geographic coordinates of the individual nodes and connecting lines in between. This is geographically correct, but simple and intuitive understanding is often more important than exact geographic references, especially in time-critical environments. Creating pleasing drawings of arbitrary networks is a well-known research area in computer science with an entire community of its own [KW01]. While general graph drawing algorithms work for any graph-based data set, they do not consider specific characteristics of the data. In particular, most methods do not retain the initial positions of nodes or directions of the edges.

Layouts for geo-referenced networks must respect both the geographic references but should also be easy to read and understand. This is an important criterion for many sensor networks such as the electric grid and other supply networks, because it enables the user to match the geographic setting with the drawing on the screen. This is summarized in the following requirement:

Requirement 6 (Relative Directions) *Preserve relative directions in node-link diagrams of geo-referenced networks.*

The main focus of the domain expert (e.g. a control room operator) of such maps obviously is the sensor network itself. Other sources of information such as weather conditions could be relevant and should be provided on demand. Clark et al. [CPF09] suggest the integration of data sources such as gas and water, but also communication structures of electric grid displays. This is required for the network analyst to assess external influence factors.

Requirement 7 (Data Sources) *Integration of heterogeneous data sources to augment sensor network data with complementary information.*

The operators who manage such a complex network need to have a precise overview to be able to act and react whenever necessary. With electrical grids growing in size and complexity, simple visualizations such as the one-line diagram quickly reaches its limits. Complexity does not only involve a large number of elements in the networks, but also different stakeholders, different data sources, etc.

More generally speaking, the analyst needs to get an overview of a sensor network in order to get an impression of the “big picture” and to identify potential problems. This requirement comprises the two requirements 6 and 7.

Requirement 8 (Overview) *Provide an overview of a sensor network in both space and time.*

In many sensors networks, a large amount of connected sensors deliver measurement readings for various kinds of data types in periodic time intervals. These events are then processed by automated algorithms such as SCADA systems. The results are then presented to the user, often in tabular lists.

Over the years, the complexity of many networks has increased tremendously. This results in larger networks per se, but also in more (event) data recordings in each of the nodes. Hay et al.

analyzed events in electric grids, concluding that only a very small subset of the data requires the operator's attention [HAB09]. However, identifying those few, but important events to get an overview on the situation can be quite laborious and time-consuming. The analyst therefore needs automated support in filtering large amounts of event data.

Requirement 9 (Automated/Manual Analysis) *Monitoring and control support for operators in order to be able to deal with large amounts of events.*

Once an area has been identified as relevant for the analysis, the operator needs to get more details to assess the situation. In particular, the spatial and temporal context must be preserved. The operator might be interested if sensors at two stations that are connected measure similar values or not. Depending on the task, the network analyst wants to know about daily, weekly and seasonal patterns and trends. In how far do the consumption patterns change over the year? What are the differences between workdays and weekends? What are the regional differences in the grid? In summary, the comparison of different sensors at the same time but also the development of a single sensor over time are both important analysis tasks.

Requirement 10 (Compare Sensors) *Comparison of different nodes but also different time frames.*

Many networks problems can be detected with automated rules such as threshold validation. However, not all problems can be explicitly stated as rule of an expert system and automatically identified. On the other hand, performing a manual analysis of all sensor readings is often hardly feasible, because it is difficult to have an eye on all sensors at the same time. Also, it is quite difficult to derive trends and frequently occurring patterns from simple line chart plots. Still, the analyst needs to be able to recognize diverging states, anomalies and suspicious patterns as quickly as possible. The similarity of individual patterns is often used as a basis to define "common" or "abnormal" patterns and trends.

Requirement 11 (Identify Anomalies) *Identify common and uncommon time series patterns in the network.*

In a final step of the workflow, the analyst must be enabled to prevent critical situations and incidents in the future. A simulation of different configurations and scenarios plays a key role in identifying the hot spots. This is required for the planning of network reinforcements.

Requirement 12 (Planning Support) *Support planning and forecast of simulated networks.*

In the following sections, we will discuss these requirements in details and provide individual contributions for each one of them. While we aim for a generic application, we will motivate the definition of tasks and goals by transfer from the specific scenario of electric grids. However, we note that these characteristics also apply to many other application scenarios such as traffic analysis, water-level-predictions on rivers or logistics.

4.3. Sensor Network Analysis

Sensor Networks form a special type of data, combining graphs and time series data. As stated in the introduction, we focus on univariate sensors that measure a single variable over time. However, we discuss possible extensions to multi-variate analysis in the conclusions chapter and refer to related work (see Section 4.6.6). We present related techniques for time series preprocessing and for visual-interactive time series analysis, which both depend on underlying data. For a discussion on graph data and visual graph analysis, we refer to the first part of this dissertation, in particular to Section 3.4.

Time series preprocessing is of great concern for the quality of almost any time series analysis application. Thus, the data mining community has spent great effort in the development of time series preprocessing techniques [DTS*08, Fu11]. Usually, pipelines are applied that may execute several cleaning, reduction, normalization, segmentation, or feature extraction steps on the underlying time series data. Recently, visual-interactive applications to support time series preprocessing and model creation have been presented [BRG*12, BAF*13]. Relevant overviews of time series visualization [AMST11] and the visual analysis of time and geo-spatial data [AAD*10] exist.

The analysis of sensor networks is often associated with a very specific application domain, be it wireless networks, surveillance systems or electrical grids. The Supervisory Control and Data Acquisition (SCADA) systems process a continuous stream from connected sensors. These sensors typically collect low-level measurement data of all kind as well as notifications and alarms across the wire to the central node. On a higher level, also intelligent electronic devices (IEDs) are connected to this network. These elements have the ability issue control commands and thus actively interact with the network.

4.3.1. Automated Event Analysis

As the network sizes continuously increases in size and complexity, the number of events has significantly increased as well [HABM08]. This lead to a number of filtering and prioritization algorithms. For example, Kyriakides et al. present a processing pipeline that also includes the generation of diagnoses and recommendation messages [KSH07]. In general, expert systems are an often-used approach to automatically process the events based on a set of pre-defined rules. These rules are usually put up by domain experts and converted into software by knowledge engineers. Using this approach consequently also permits to identify root causes of a larger set of error messages. A short introduction to this field is given by Rooney and Heuvel [RH04], a more in-depth analysis is given by Julisch [Jul03]. The complexity of this rule database can quickly become rather extensive and thus cumbersome to maintain. Marques et al. therefore present a knowledge-based system that work with simpler fuzzy logic rules [MTF05].

Inducing more domain knowledge into the system permits to perform monitoring and problem analysis on a higher level. Using semantic reasoning, sensor data can be put into spatial and temporal context allowing for complex queries [CHL*09]. The world-wide-web consortium recently published a formal ontology specification of sensor network data [CBB*12]. Li et al. advocate

intelligent alarm management based on expert systems to be able to cope with the ever increasing amount of alarms [LQS*10].

Basically, an expert system contains a set of domain-specific facts, rules and an inference engine that derives new information from this knowledge base. The setup process that transforms the knowledge of domain experts into its adequate machine-readable representation is a complex and time-consuming task known as *knowledge engineering*. Software engineers can perform this operation either manually or with the help of domain-specific languages (DSLs) which are still machine languages, but closer to natural language and the user's domain.

In addition to the rule definition, the inference process must be specified. In its standard form, it comprises one or more premises – the condition – with a conclusion that is evaluated if all premises hold. These conclusions can insert, update or retract facts from the knowledge base. This modification leads to another evaluation of the rules' conditions and fires them where necessary. Inference engines commonly have two modes of operation:

- Forward chaining: Iterating over every rule, the engine tests the conditions with all known facts. When all conditions are fulfilled, the rule fires and the conclusion is drawn.
- Backward chaining: Starting from the opposite side with a set of hypotheses the engine tries to find facts that support these assumptions.

Some systems also support “mixed chaining”, a combination of both approaches while others are able to involve external information sources to query for additional knowledge.

From the monitoring and management perspective, we split existing approaches for the analysis of event data with respect to the time frame that is used. Typical analytical approaches focus on rather long-term knowledge building, while monitoring approaches often aim to support short-term situational awareness. A monitoring and diagnosis system for time-series data has been presented by McLachlan et al. [MMKN08]. We adopt their drill-down approach to diagnose the current state of nodes in a network. For clinical data, Shahar et al. propose an exploration system that features a knowledge base to derive a patient's status from status measurements [SGBBT06]. Similarly, Torralba-Rodriguez et al. [TRFBMBM10] work with a knowledge base of medical data and a custom-tailored evaluation language. Their rule-based system creates diagnoses for a patient's symptoms, but is also able to explain why a certain diagnosis has been put up.

All of the presented approaches focus on early detection of received event data. This might be already too late as initiating countermeasures also takes some time. In particular when cascading effects could occur, early detection combined with fast response times are key. Simulating the development of measurement readings can help estimating the system state in the near future to buy additional response time. Surprisingly, there is not too much literature available on that topic [OMWC05]. One noteworthy exception is the work of Li and Tate who simulate forecast errors for wind turbine power generation [LT11]. The authors show their results in a line chart that is enriched with uncertainty information.

4.3.2. Visualization Systems

One shortcoming of automated approaches is that the user has to rely on its functionality without the ability to visually verify the results. As discussed in Section 2.6, integrating the user in one or more steps of the analysis pipeline can increase the overall quality of the process. Visualization is an important measure to convey information to the user. Using interactive tools, the user is enabled to interfere and tweak the automated algorithms. Combining the strengths of machine-based computation and human understanding promises to lead to improved analysis process, in particular for explorative tasks.

We locate our contribution between knowledge-assisted visualization following the definition by Chen et al. [CEH*09] and, more related to the application domain, the monitoring and management of complex systems. In knowledge-assisted visualization, a knowledge base is added to Card's well-known visualization pipeline [CMS99a] to control the visualization. Using this knowledge representation, [MHS07] et al. and Gilson et al. [GSGC08] select suitable mappings from the data and its schemata. Kohlhammer used knowledge representation for decision-centered visualization [KE05], focusing on situation awareness in emergency management. Going deeper into the internals of the reasoning process, Shi et al. [SQW11] display the internal reasoning network of a Rete-based expert system with an interactive animation. Possibly most related to our work, Garg et al. [GNRM08] and Xiao et al. [XGH06] combine data visualization with an interactive construction of its underlying models.

The display of situational context has been investigated by Laufenberg [Lau05]. A web-based monitoring tool has been presented by Clark et al. [CPF09] that aims to integrate additional data sources that is designed for suppliers, but also for customers. With a focus on the flow and transfer capability of power lines in large-scale systems, the work of Overbye et al. [OMWC05] sheds light on different visualization techniques. In collaboration with Overbye, Klump et al. present a visualization system that is specifically tailored to industry-demands [KSO02] and reports on practical experiences in the domain. To the best of our knowledge, knowledge assisted visualization has not been applied to the smart grid monitoring domain in the approaches we discussed. Li et al. [LQS*10] state that the major task of visualization is to provide situational awareness through integration and appropriate display real-time state variables. Some of their goals have already been realized in our system. From an engineering perspective, the control room visualization tools are at the end of an event processing pipeline. When it comes to management, typical systems display event data in tabular form. The operators that monitor the system need to manually acknowledge these messages.

Some visualization systems deal with the analysis of time-series data more explicitly. Stoffel et al. present a client-server visual analytics systems for anomaly detection in computer networks [SFK13]. Its main views show a collection of vertically oriented line charts that are compared with a reference model of the data. An inspiring technique is the calendar view [VWVS99] by van Wijk et al. Similar to our approach clustering of daily patterns is applied to visually encode a calendar visualization. However, the original calendar view differs from ours in the chosen clustering technique and the color coding (which is not similarity-preserving). A technique that combines the comparison of daily temperature patterns and geo-spatial meta data information was presented in the digital library context [BRS*12a]. However, a chronological representation of the

daily patterns is not provided. Techniques that focus on the visualization of periodic time series data may rely on radial [ZFH08], cyclic [TS08], or on projection-based layouts [BWS*12, WG11].

Other visualization systems focus more on the network part or the spatial environment in general. The GreenGrid visualization system of Wong et al. focuses on the analysis of very large electrical grids [WSM*09]. It uses geographic node references to provide a ground truth coordinate system, but displays the same grid in a distorted view to make better use of the available screen space. Similar to our approach, it illustrates the potential advantages of force-directed layouts based on physical properties over geographic coordinates. The visualization system of Hadlak et al. analyzes the temporally changing link quality of a wireless network [HSCW13]. The authors cluster time-series data and display the data in a node-link diagram, but we take this one step further and enable the analysis of repeating patterns which was not part of their work. It also allows for interactively comparing different clustering settings, together with a tree view of the time series data. Saraiya et al. conduct a user study to evaluate different node glyphs for graphs with multi-variate node attributes [SLN05], which we use as a guideline for our glyph design. Using a combination of spiral visualizations and treemaps, Janetzko et al. detect anomalies in power consumption data of commercial buildings [JSMK14]. Shi et al. demonstrate anomaly detection for multi-variate sensor data in hierarchical networks. In contrast to our work, the authors do not focus on pattern analysis [SLH*11].

The last group of related work are explicitly settled in the electric grid application domain. The report of Overbye et al. [OW01] gives a broad overview over a range of visualization techniques specifically for substations and the power flows in between. Overbye's application domain is the energy transmission in large-scale grids. Some of their concepts have been evaluated through user studies [OWRS03] and transferred to practical applications in the industry by Klump et al. [KSO02].

Most visualizations work with a static view of the grid that is annotated with status or flow information data [OWL99]. Evaluation has shown that dynamic resizing and appropriate coloring of visual elements in critical condition attract the user's attention [OWRS03].

There are visualizations that deal with the sensor networks on the smallest scale possible – the device level. For example, Meliopoulos et al. simulate and visualize electro-thermal model visualization of generators & transformers to analyze the health status of devices in the electric grid [MCO04]. Going one step further, 3D visualizations have also attracted researchers interest. Above all, one additional degree of freedom is available for visualization. This axis can be used, for example, to augment plain 2D visualization with additional information [SO04]. Typical elements are stacked cylinders that represent some data quantities; more complex glyphs can provide even more information. According to Overbye et al. [OMWC05] this leads to a faster detection of violations. On the downside, these visualization tend to be less intuitive and occlusion of important elements can occur. Also, perceptual disambiguities of size with respect to the viewing distance is a problem.

The monitoring of power systems is related to visualization concepts, but somewhat different in certain aspects. Clark et al. present a map-based web tool for operators but also for end-users for consumption analysis [CPF09]. We picked up on their idea of integrating tangential infrastructures other resources to achieve synergy effects.

4.4. Network Layout Strategies

Visual representations of network structures have been studied extensively, with various specializations for different use-cases. Our focus is on supply networks with nodes that correspond to physical entities, and their properties that change over time. Similar to our network visualization, Hadlak et al. use embedded line charts to link time series data to graph nodes [HSCW13]. An overview of the design space for temporal graph visualizations can be found in the work by Kerracher et al. [KKC14].

The survey of Cockburn [CKB08] discusses several aspects of the user's focus in graphical interfaces. It also discusses *semantic zooming*, a technique where the level of detail of the visualized entities corresponds to the zooming level. Applied to map navigation tasks, it reduces the task completion time and we therefore employ it in our visualization system.

In this section we analyze different layout algorithms that preserve relative directions in geo-referenced networks. Even today, the layouts of these networks are often created manually. This is due to the requirement that these layouts must respect geographic references but should still be easy to read and understand. The range of available automatic algorithms spans from general graph layouts over schematic maps to semi-realistic drawings. At first sight, schematics seem to be a promising compromise between geographic correctness and readability. The former property exploits the mental map of the user while the latter makes it easier for the user to learn about the network structure. We investigate different algorithms for such maps together with different visualization techniques.

In particular, the group of octi-linear layouts is prominent in hand-crafted subway maps. These algorithms have been used extensively to generate drawings for subway maps. Also known as *Metro Map* layouts, only horizontal, vertical and diagonal directions are allowed. This increases flexibility and makes the resulting layout look similar to the well-known subway maps of large cities. The key difference to general graph layout algorithms is that geographic relations are respected in terms of relative directions. However, it is not clear, whether this metaphor can be transferred from metro maps to other domains. We discuss applicability of these different approaches for geo-based networks in general with the electric grid as a use-case scenario.

4.4.1. Types of Network Representation

We group layout strategies into three different categories based on node position constraints: *General Graph Layouts* with no restrictions on node positions, *Schematics* which assume some degree of correlation and *Topographic Maps* which stick to geographic locations rather strictly (see Figure 4.1).

Force- and energy-based layout algorithms interpret the graph data as a set of nodes that is connected by virtual springs. Based on different energy constraints, these springs and other actors move the nodes into a state of minimal energy. While this is a natural way of implementation, the drawing quickly becomes tangled up if the number of nodes becomes too large. Topographic maps, on the other hand, aim to create a picture based on a set of reference positions (see subsection 4.4.4.5). Such displays aim to exploit the user's mental map of the environment and are therefore often used for navigation (e.g. road maps).

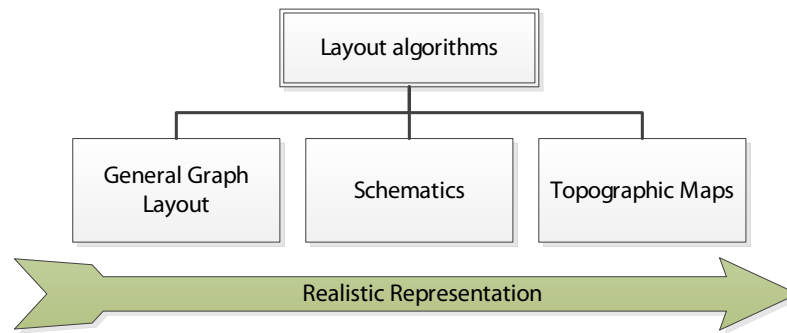


Figure 4.1: Three representation categories with increasing degree of realism from left to right.

Projection-based methods on the other hand, first create a layout in a high-dimensional space and project it according to different quality metric into 2D space. They try to find a globally optimal solution and do not suffer from scaling issues, but the results are often not very intuitive. The position of individual nodes is meaningless, because it is derived only based on a defined node distance metric.

The rationale behind schematic maps is to remove information from topographic maps that is irrelevant for the task the user is trying to solve. Using simplification algorithms, unnecessary details are removed, leading to a clearer, abstract picture. Such information is, for example, the geographically correct position of line segments. Schematics are an (topologically equivalent) abstraction located between topographic maps which use with fixed node positions and traditional graph drawing that do not pose any constraints on vertex positions [CdBvK05]. Figure 4.2 gives an overview on different abstraction levels for geo-based networks.

One of the first attempts to create such a schematic map was a drawing of the Metropolitan Railway London from 1896 which used distortion to better exploit the available drawing space. Among the most popular schematic maps is one created by the British engineer Harry Beck, who designed a layout of the London Underground in 1933. This map is based on stratified edges so that they are easier to follow and restricted angles to multiples of 45 degrees. Consequently, only eight directions are available for layout and these maps are today known as *octi-linear* layouts. The book of Garland contains not only an introduction into the topic, but also aesthetic criteria and rules for the manual creation of these maps [GB94]. Examples for manually drawn contemporary designs can be found, for example, in the work of Morrison or Ovenden’s book [Mor96, Ove05].

“If you’re going underground, why bother about geography? It’s not so important. Connections are the thing.” [Harry Beck]

In course of the study performed by Dwyer et al. on graph layouts, many participants preferred generated layouts over hand-drawn ones. They chose layouts “based on general aesthetics, most commonly a symmetric, ordered, or clean look” [DMW09]. Nonetheless, creating such layouts in an automatic, computerized fashion is a challenging problem [Tuf97]. Nöllenburg was even able to prove that the decision whether a given graph can be drawn using an octi-linear layout or not is a NP-hard problem [Nöl05].

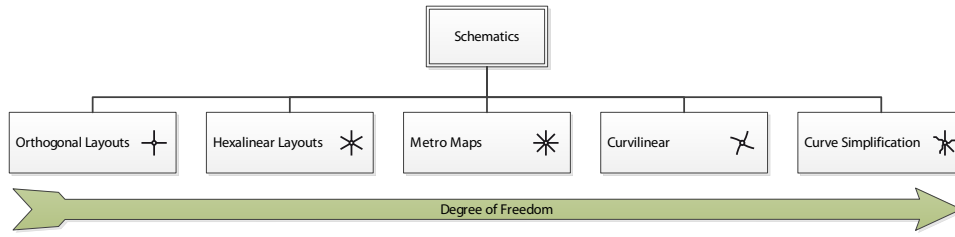


Figure 4.2: *Different groups layout algorithms, organized by the freedom of vertex positions.*

4.4.2. Design Space Criteria

The number of approaches that create schematic drawings in one way or another is large. We therefore define a collection of general constraints that can be used to evaluate the drawing quality of the different approaches. For a list of layout criteria of general node-link diagrams of networks, we refer to the work of Tollis et al. [TBET98]. Searching for a more precise definition of quality in the context of octi-linear layouts, Hong et al. studied different manually created diagrams [HMN05]. The authors do not focus on geographic or topological correctness, though. Other rule sets can be found in later publications [HMdN06, SR05], the most comprehensive one in the survey of Wolf [Wol07]. This set of design criteria has been put up for the design of subway maps, which we adapted for schematic drawings in general. We therefore abstract the rules and summarize them as follows:

- Preserve the topology of the network
- Restrict the number of allowed directions
- Ensure a minimum distance between nodes
- Minimize the total number of kinks in edges
- Preserve directions (relative positions) of nodes
- Minimize the total edge length to compactify the drawing
- (Display individual lines as such)
- (Align text labels so they don't overlap)

We put the last two rules in brackets, indicating that this rules are not relevant for all network types. Labels are important for orientation purposes, but requirements for placing, etc. can be very different from those for metro maps. This topic is further discussed in subsection 4.4.4.1.

These six defined criteria are of rather aesthetic nature, not a set of fixed mathematical constraints. Running an optimization against one of the variables might lead to a decrease in quality for another. The problem is a multi-objective problem that has a potentially large number of

Pareto-optimal solutions. Which one “looks best” is thus up to the user. To give an example, Nöllenburg and Wolff separate rules into hard and soft constraints. Hard constraints *must* be fulfilled at all costs, while soft constraint are desired, but are not critical [NW11].

To evaluate the usability of a certain algorithm with respect to a given scenario, differences between the application domains must be identified. For example, metro stations generally have a similar purpose. In other networks, however, different roles may need to be represented by the nodes of the network.

One of the most important properties is the size of the network. Metro maps are rather small, compared to many geo-based networks which can have thousands of nodes and edges. For example, with roughly 300 stations connected by roundabout 400 line segments, London has one of the largest subway networks in the world.

Another aspect is the structure of the network. In many hand-drafted metro map drawings, the distances are distorted to achieve a more compact drawing. Down-town stations often get more space to avoid overlaps while suburban areas are shifted towards the city to reduce the overall size. This is also useful for the layout of other networks, in particular large ones with heterogeneous areas. Densely populated areas typically contain more lines than area with lower population density. Metro networks often have a single, densely packed center area surrounded by suburbia in a circular manner. In general, no a-priori knowledge on the network density is available.

Also, metro lines do not have an inherent hierarchy that could be exploited for aggregation. In contrast to that, many man-made infrastructures are often designed in a tree-based fashion. For example, the street networks in urban areas contain different types of streets that are often organized hierarchically. Similarly, supply networks such as the electric grid exhibit such a structure. Technically speaking, the grid contains lines that operate at different voltage levels. The high-level transmission grid moves the produced energy in overhead power lines over large distances across the country. One level lower, the so-called *sub-transmission* connects the transmission and the distribution grid, which operates at the lowest level. The voltage is gradually stepped down on each level until the electricity reaches individual commercial and residential consumers. In contrast to that, metro maps are typically based on a set of individual lines that reach from one endpoint to the other, often across the entire city. This also implies that multiple lines run in parallel, in particular at the center of the map. It is important that passengers can easily differentiate between those individual lines. Connections in the supply networks on the other hand are typically limited to consecutive stations. This makes it easier to find connections, because the idea of “changing trains” is not relevant here.

4.4.3. Layout Algorithms

In this subsection we present an overview on different layout algorithms. We distinguish techniques by their degree of freedom with respect to node positions. Starting from the the group with the highest degree of freedom, we will iterate until we reach layouts with fixed vertex positions. See Figure 4.2 for an overview.

4.4.3.1. Orthogonal Layouts

We start with orthogonal layouts, an approach that originates from chip design. These chip designs are typically very large necessitating fast layout computation times. Another constraint from that domain is to avoid edge (i.e. wire) crossings at all costs. Nodes are restricted to only four orthogonal directions, resulting in an embedding of a 4-planar graph. One advantage of orthogonal layout is their symmetry which motivates the usage of such diagrams also for organizational and flow charts. Also, parallel lines can be identified as such and adjacent edges can be distinguished easily. Often, node coordinates are also restricted to integer grids, which further reduces the degree of freedom. Consequently, many edge kinks are required to satisfy these constraints. This makes it harder to follow edges and increases the total edge length [EFK01].

The drawing of orthogonal layouts can be efficiently solved using the *topology-shape-metrics* algorithm [Tam87]. The graph is converted into a directed, acyclic graph (DAG) with one source and one sink. Computing the Minimum-Cost-Flow problem finds the path with the lowest cost of the flow between two nodes. Here, cost refers to the number of bends. While this method also works for hexa-linear layouts (with 60° angles), it fails for smaller angles.

Also coming from background in electronics, so-called *one-line diagrams* are often used for the analysis of power flows. Similar to orthogonal layout, directions are often limited restricted to four, but sometimes up to eight different directions. Nodes are displayed used standardized, domain-dependent symbols representing electronic devices. Klump et al. present a visualization tool with a focus on one-line diagrams [KSO02]. See subsection 4.4.4 for different visualizations based on one-line diagrams.

4.4.3.2. Hexa-linear Diagrams

Using 60° angles between directions increasing the degree of freedom from four to six. The rather strict limitations of orthogonal layouts are relaxed. In these *hexa-linear* drawings, however, no right angles are allowed. This can have a negative impact on the symmetry of the drawing [Rob12]. Using twelve instead of six directions (at 30° angles) overcomes this problem, but makes the drawing increasingly complex.

The approach of Merrick and Gudmundsson is able to restrict the number of allowed directions to six, but does not ensure the preservation of relative directions [MG07]. To the best of our knowledge no automatic generation algorithm for the creation of direction-preserving drawings exists to date.

4.4.3.3. Metro Maps

Going one step further, octi-linear layouts allow for eight different directions. The following three algorithms that we briefly outline have already undergone an extensive analysis in the past. For more details, the interested reader is referred to the article by Wolff [Wol07].

In the work of Hong et al., a force-directed layout is used that is similar to those for general graph layout [HMN05, HMdN06]. The initially geographical map is converted into a set of repelling nodes. Virtual springs are inserted for each pair of connected node, which attracts the two linked nodes. Additionally, imaginary forces (the authors call them “magnetic” forces) rotate

edges so that they align with one of the eight allowed directions. This is a local optimizing approach that looks at one node at a time. Based on geographic layout, Wong et al. apply a modified version of their general-purpose force-directed multilevel layout algorithm [WSM*09]. Different attributes are mapped to node repulsion forces and edge lengths to highlight specific properties of the grid. Due to its multi-grid architecture, this approach is one of the fastest.

Stott and Rodgers propose a method that uses *Hill Climbing* to solve a defined multi-criteria problem [SR05, SRMOW11]. The algorithm first maps all nodes positions to a fixed grid. An objective function based on a set of aesthetic rules defines the quality of the drawing. The hill climbing algorithm then iteratively applies random changes on one node position at a time and re-evaluates the objective function. If the value is higher than before, the new position is accepted. The cycle is repeated until no further improvement can be made. This is a rather simplistic approach that gets stuck easily in local minimum, failing to find the best global solution. The authors propose a fix to mitigate this shortcoming, but the basic problems remains.

A different approach has been proposed by Nöllenburg and Wolff who use linear programming to find a globally optimal solution [NW06, NW11]. Trying to find a solution for all nodes at the same time is in strong contrast to the force-based approaches. It is also quite different from typical linear programming methods. The key difference is that it uses both real-valued and integer variables, resulting in mixed-integer programs (MIP) which require dedicated solvers. The authors justify the use of such heavy-weight solvers with the complexity of the problem. The use of integer variables and constraints renders the solution space finite, but finding a solution becomes more difficult. This might be counter-intuitive at first sight, but can be explained by the fact that geometric properties of real-valued optimization strategies cannot be transferred into integral space. Often, not all requirements can be satisfied. While the general idea of the approach is in some aspects similar to that of Stott, the complexity of the rule base and the solver are significantly higher. This fact is also reflected in the computation times.

4.4.3.4. Curvilinear Layouts

Fink and the authors of the MIP approach present an extension of octi-linear layouts, namely *curvi-linear* layouts [FHN*13]. The authors use Bézier splines instead of straight line segments to represent connected nodes. A striking advantage of that approach is that no kinks are needed to change directions which makes it easier to follow lines. Fink even advocates avoiding edge kinks at nodes.

This approach is motivated by a study of the Paris metro plan by Roberts et al. [RNL*13]. Compared to conventional metro maps, users were able to plan their journey significantly faster using a hand-drawn curvilinear map. Roberts argues that the octi-linear layouts were not necessarily ideal for route planning tasks. The kinks that are created by forcing the layout to become octi-linear also introduces cognitive load on the user.

According to Fink, curvilinear layouts are particularly useful in scenarios where networks are very dense or the trajectories of individual transportation lines are very complex. The authors also present a promising approach that allows for automatic generation of such maps. In their approach every segment is a spline which leads to potentially many S-shaped elements which are difficult to follow. Also, the placement of labels was not yet realized.

4.4.3.5. Curve Simplification

The Curve Simplification category gathers different approaches that are less strict in terms of topology. Many of them aim to produce octi-linear layouts, though. Barkowsky et al. use a curve simplification algorithm to reduce the complexity of a given geographic layout [BLR00]. The authors do not, however, constrain the line's orientation or the distances between nodes. Avelar et al. align edges of a given network in eight directions so that a set of aesthetic criteria is fulfilled [AM00]. The algorithm computes node positions iteratively based on the position of neighboring nodes. This ensures that the relative positions are retained, but it is not always possible to create octi-linear edges for all nodes. The labeling of nodes is not considered, though. This work has been enhanced by Ware et al. with a focus on mobile devices [WTAT06]. Similar to Barkowsky et al, Merrick and Gudmundsson propose a path simplification algorithm [MG07], dedicated for the automatic creation of metro map layouts. Since the algorithm processes paths separately, the relative directions are not preserved in general.

Cabello et al. create a schematic map based on a given graph layout without adjusting the node positions [CdBvD*01]. Edges are simplified similar to the approach of Barkowsky et al. but also constrained to certain fixed angles. This makes it one of the few strictly octi-linear methods. As the nodes remain at their original position, the effect of the simplification is somewhat diminished, however. The authors demonstrate in a follow-up publication an approach that also allows shifting of nodes so that they align on horizontal, vertical and diagonal axes [CK03]. This is done, however, at the cost of relative directions.

4.4.3.6. Summary

Most publications contain one or more use cases and explicitly state computation times for data sets. Many of them use actual metro map data for their setup, in particular the Sydney metro network. In general, however, different data sets and different machines were used. Some approaches also include the computation of labels in their algorithm. Consequently, the timings *must not* be used for direct comparison. They indicate an order of magnitude, though, which allows the reader to get a rough idea of the time complexity.

An effective measure to reduce computation times that is used by different authors is to temporarily simplify the network. The complexity is reduced by removing nodes with only two edges (i.e. those that lie on a single line). These nodes are re-inserted afterwards at estimated positions along the line. This can speed up computation with only little loss of layout quality. The spring embedder and the hill climbing algorithm tests were performed on such a reduced data set, which contains only a fraction of the original data (less than 25%). Similarly, the authors of the MIP approach also work with a simplified network, which is about 50% of the full graph data set. As a rule of thumb, there is always a trade-off between quality and speed when it comes to selecting the right algorithm. Consequently, the choice depends on the task and the use case. Among the fastest algorithms are force-direction approaches such as Hong's spring embedder [HMdN06] and the multi-level layout algorithm by Wong et al [WSM*09]. These approaches are fast enough for interactive or almost-interactive applications. They could also be used to create a preview that

gives a first impression of a large network. If time is not a critical constraint, slower, MIP-based solutions can be used to create high-quality results.

Some algorithms can return a valid solution at any time, i.e. before the simulation terminates by itself (i.e. *anytime algorithms*). For example, Nöllenburg and Wolf report that their approach yielded a solution that was fairly close to the final solution when only 10% of the time had been elapsed. In sharp contrast to that, incorporating node labels (see subsection 4.4.4.1) increases the running time of their computation from 30 minutes to almost five hours.

The number of allowed directions is crucial for the decision of the right algorithm. Merrick and Gudmundsson [MG07] put layouts with 4, 6, and 8 valid directions in juxtaposition. With increasing number of allowed directions, the drawings resemble more and more the original data set. We conclude that octi-linear layouts create simplified drawings while still preserving the original structure of the data. From the set of discussed categories, we will therefore limit ourselves to these layouts.

We also assume that the network topology does not change too often. In such cases, the layout computation can be computed once and reused later on. This allows us to use high-quality layout as a basis for visualization concepts. Most of the publications that focus on visual interaction concepts use MIP-based layouts.

4.4.4. Visual Interaction Concepts

The visual design of nodes and edges is a part of the design space, which is – to a large degree – independent of the placement of nodes and edges. The means of visualization is to convey the information in the data to the perception of the user. The usability of a visualization technique depends on the task the user wants to solve. For example, traditional metro maps are designed to optimize route planning. However, due to the distortion of the map, it is hardly possible to estimate distances and travel times. Details on the useability of metro maps can be found in the work of Roberts [Rob12, RNL*13].

4.4.4.1. Node Labeling

Using labels is probably the most prominent approach to support the user's orientation in the diagram. Imhof gives an overview on different criteria for node labels [Imh75].

We identified two ways to classify node labeling algorithms: One of them differentiates between those who take the node labels into account during the layout computations and those who add them to an existing layout in a post-processing step. For example, Stott's *Hill climbing* and Nöllenburg's *MIP* algorithms include the label placing in the layout algorithm while Hong's *Spring Embedder* places labels afterwards. When the label (i.e. text size) is used for the layout computation, the text cannot be changed afterwards. This could be an issue if the label is supposed to reflect live status information.

The other classification differentiates intrinsic and extrinsic labeling: Intrinsic labeling is about placing the label sufficiently close to the point they refer to (this point is also called *site*). External labeling places the label further away (where more space is available) and links the label to its site with a (straight) line segment (the *leader*).

Wu et al. demonstrate a sophisticated external labeling approach by reserving enough space for small images of metro stations in the maps [WTH*13]. Fekete and Plaisant demonstrate an approach that displays labels in a circular area around the cursor [FP99]. Labels are placed outside this area and lines connect them to the point they refer to. Similarly, Bekos et al. also demonstrate an approach for external labeling, but place all labels at the border of the drawing [BKSW07]. The *LabelHints* approach of do Nascimento et al. aims at finding the maximum independent set of non-overlapping nodes [dNE08]. It is a map labeling system based on user interaction to prevent overlaps between labels. Apart from the two labeling approaches, some authors place labels *inside* the node glyph. Due to the different text lengths, some of the labels are truncated. See Figure 4.3 for an illustration.

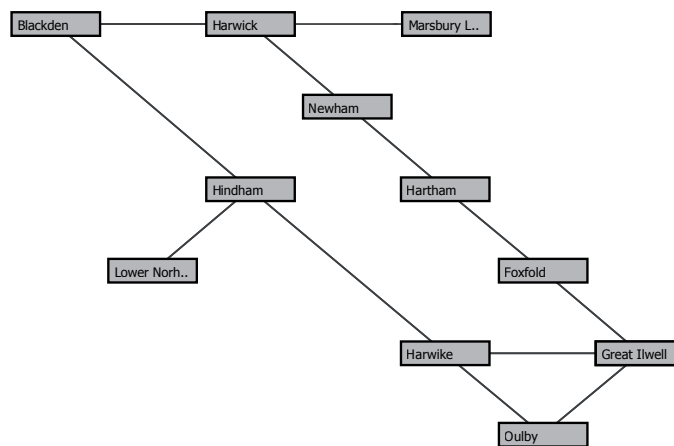


Figure 4.3: An octi-linear layout of a substation network. Labels are placed inside the nodes and truncated where necessary.

4.4.4.2. Node and Edge Display

In conventional metro maps, node glyphs are classified into two groups. Stations with only two directions are represented by a simple tick mark across all passing lines. Stations with more than two are typically displayed as circles or polygons (see Figure 4.4). This is different from typical graph drawings, because of the large number of nodes with degree two.

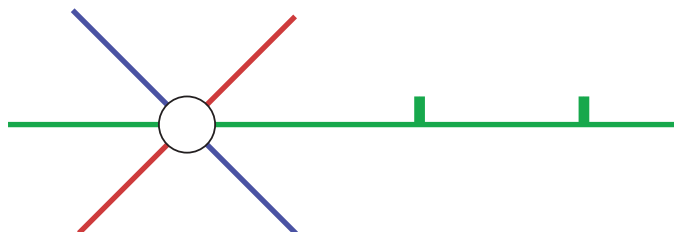


Figure 4.4: Minor stations are indicated by tick marks, larger ones by circles or polygons.

The size of the elements can be used to intuitively represent general importance, centrality and/or size. Overbye et al. discuss different visual representations to encode physical quantities in the display [OWL99, SO04, OMWC05]. One of them is animated arrows along edges to illustrate energy transfer. A user-study indicates that they support the user in determining power flow patterns better than digital displays, but also increase visual complexity. In another illustration, pie charts and gauges were used to indicate workloads. The authors also experiment with 3D shapes such as cylinders and cones to convey additional information to the user.

4.4.4.3. Focus plus Context

In contrast to physical maps, computer visualizations can also incorporate interaction techniques to adjust the display to the user's needs. One of the main tasks of visualization is adaption to networks that are larger than available screen area. One typical approach is to use panning and zooming using a virtual camera. While this allows the user to navigate, only a small part of the entire network is visible, which impairs the user's orientation.

Different *Focus plus Context* techniques such as fish-eye views can be used to emphasize some areas while preserving at least some information around it. Often used attribute modifications are size, level of detail and color intensity. Different scaling techniques for graphs have been investigated by Carpendale et al. [CCF97]. Haunert and Sering analyzed different Focus & Context methods with a focus on road networks [HS11]. A major problem in most networks is that edge crossings can be introduced by non-linear scaling of features.

An alternative to separating the view into distinguishable focus and context areas is to create a transformation between two views. Using time, Reilly and Inkpen present an approach for interactive warping between topographic and octi-linear map [RI04, RI07]. Böttger et al. demonstrate a similar, automatic approach to distort a topographic map so that it fits into a metro map layout [BBDZ08]. Using an interactive fish-eye effect, the map is distorted to suit both the metro map and the geographic layout. This allows the user to get geographic information such as the location of the airport. The amount of distortion can be adjusted from geo-graphic to octi-linear layout. This allows the user to correlate metro stations with over ground environment. However, due to the distortion, it becomes difficult to correctly estimate distances.

Traditionally, the task is route-planning for metro maps, but the analysis of connected paths is equally interesting in other domains. The basic idea is to display a particular path in focus, arranging the rest of the network around. This path in focus is displayed as straight as possible to make it easier to follow. The rest of the network must be transformed accordingly. Wang and Chi [WC11], present a fast, force-based algorithm with interactive response times to accomplish that. In their approach, the path in focus is octi-linear while the context network is created by curve simplification.

4.4.4.4. Spider Maps

One approach to preserve the context of a selected region is to use *Spider Maps*. They are a non-interactive *Focus plus Context*-based visualization, with different layouts for the focus and

the context area. These route maps have been specifically designed for bus traffic¹. A Spider Map combines geo-referenced topographic map of the focus, surrounded by a schematic map of more distant destinations. One or several closely located bus stations are put into the center of a conventional road map. The road map is annotated with street names, bus stops and important buildings. This area is surrounded by a metro map layout of the wider area. The intention is to make it easier for passengers to find the right bus stop. For longer distances, passengers are interested in general directions, direction changes, etc. and not so much in geography. Landmarks are added to both views as anchor points for orientation.

Spider maps are more difficult to realize than metro maps, because additional criteria have to be fulfilled. The nodes in the metro map must be located so that they are not obscured by the road map. Also, it must be ensured that roads crossing the border are continued smoothly. Typically, spider maps do not display the entire network. They are rather focused on a single element and its connections. Hadlak et al. present an interactive approach for integrating different network visualizations into a geographic map, but does not include schematics [HSS11].

4.4.4.5. Other Displays

Pure topographic maps do not use the schematic context. They work with the actual geographic coordinates for the entire the data set. Mittelstädt et al. work with a de-saturated map in the background providing rich geographic context [MSS*13]. The authors display electric grids using symbolic glyphs connected with edges that were created using curve simplification. Nodes in dense areas are aggregated to avoid overplotting. This setup is illustrated in Figure 4.5. In their overview visualization, Overbye et al. omit the display of a map and work on a white canvas [OMWC05]. The authors connect nodes with straight line segments and reference larger geographic regions by outline and text in the center. In the approach of Wong et al., geographical references are used for the initial setup of the layout. Additional information can be encoded using different color ramp in both nodes and edges [WSM*09].

4.4.5. Discussion

This section discussed in how far different schematic layout algorithms can be used for the display of geo-based supply networks. Each of the techniques performs a trade-off between realistic representation and readability. On the one hand, a realistic representation exploits the user's mental map of geographic circumstances. Increased readability, on the other hand, makes it easier for the user to grasp the structure of an unknown network. Finding the optimal trade-off between the two therefore depends on the user and the task.

In general, we consider the restriction to eight directions (i.e. octi-linear layouts) to be a sound compromise. These layouts create a simplified schematic drawing of geographic networks, while retaining the relative position of the nodes. Compared to other geo-based maps, this trade-off is favoring readability, while preserving relative directions. This preserves the mental map of the user who is already familiar with the geographical map. Apart from subway maps, these layouts have already been successfully applied in other traffic-related domains, such as bus and train

¹<http://www.tfl.gov.uk/gettingaround/maps/buses/>

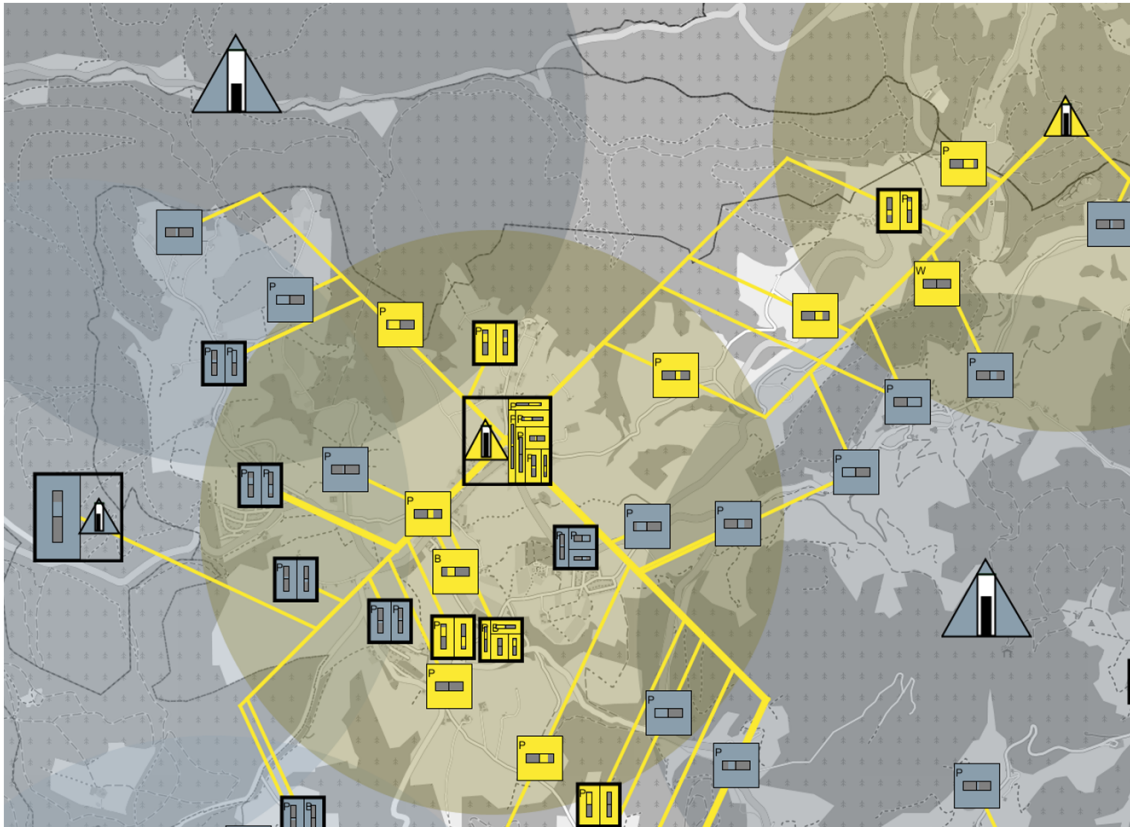


Figure 4.5: *Electric Grid on a topographic road map. Used with permission [MSS*13].*

maps. However, there are also a few visualizations in very different contexts, for example the display of abstract knowledge [Nes04, NMPR07, SGH12].

The size of the network (i.e. the amount of nodes) makes working with the entire data set impractical for the user. Also, algorithms become slower, force-directed algorithms create tangled drawings. For example, the number of nodes should not exceed a few hundred nodes for MIP. As the number of constraints increases, solution finding becomes slower or even impossible. The network complexity can be reduced by exploiting the inherent hierarchical nature of most supply networks. No clustering algorithm is required, because hierarchy layers of the data can be used. In case, that the data does not inherently contain such a hierarchy, the geographical information itself can be used to create partitions, for example along jurisdictional boundaries. We therefore consider hierarchical layouts such as the multilevel approach of Wong et al. as promising approaches. This hierarchical nature also motivates using hierarchical visualizations.

Local changes should have a local effect only to preserve the mental map. To the best of our knowledge, this property is not fulfilled by any of the currently available algorithms. There is also no study yet on how (local) changes affect the generation of such layouts.

The user could be allowed to change the focus to get details for a particular node in the context of the entire network, based on a subgraph with limited distance around a user-defined focus node. This focus area could be created using a high-quality layout while the larger context area around is created by fast curve-simplification. While this might not be suitable for overviews, it could be useful for a drill-down analysis such as fault detection. May et al. propose an approach for navigation in such locally-bound views [MSDK12]. Arrow-shaped landmarks at the border of the visible area direct to interesting parts of the network. Such visual cues could be used to point to geographic landmarks. This display could be adjusted depending on the zoom level.

As the name implies, these octo-linear layouts permit only eight different directions. Consequently, nodes with more than eight neighbors cannot be handled properly. Analogue constraints exist for orthogonal and hexa-linear layouts. One idea is to give up the constraint for these nodes and permit all directions. Another one is to split these nodes into multiple, smaller ones. Splitting basically gives 2×7 directions plus one connection between the two. However, four directions are immediately crossing (see Figure 4.6) and splitting thus gives only two additional directions. Although the number of directions is not strictly limited to eight, the number of nodes with a higher node degree should be small. In hand-crafted drawings, this problem is often addressed by inserting large rectangular nodes that span across several parallel edges. Up to date, no algorithmic placement for such special nodes exist. Edges that are meant to go into the same direction, but to different nodes impose a similar limitation. This, however, can be countered by the insertion of dummy nodes.

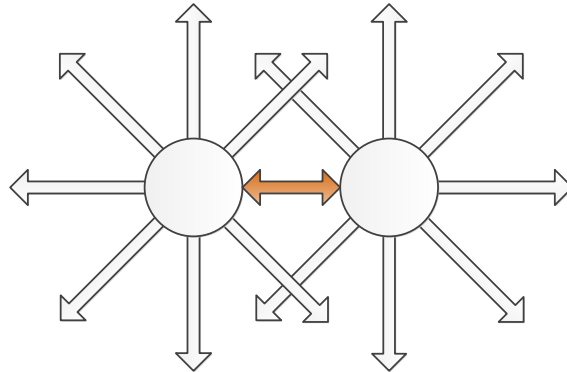


Figure 4.6: *Splitting a node into two gives six additional directions, but four of them are immediately crossing. Therefore they cannot be used to connect more nodes.*

Similar to landmarks, major elements are useful for orientation and therefore should be annotated. Node details such as labels are not necessary for all nodes, in particular not for small elements. We propose to show labels on demand, for example in tooltips that appear when the mouse cursor hovers of a node, similar to the approach of Fekete [FP99]. In hierarchical displays, labeling could be shown depending on the zoom level. This could be useful, especially if the node density is similar across the entire data set.

During design, it is important to balance the visual complexity and the amount of encoded information. If more than one variable is recorded per node, the node glyph would need to aggregate

this information accordingly, similar to the approach of Mittelstädt et al. [MSS*13]. In monitoring environments, this information is also expected to change over time.

Being easy to identify, landmarks are anchor points for orientation. The user tries to match the mental map of the area with the given schematic map. Natural elements such as rivers and coast lines, but also man-made structures are therefore often used as landmarks. There is not yet an generative approach for placing landmarks in such schematic maps. Therefore, manual improvement of computed maps is required. For interactive maps, however, the automatic arrangement of landmarks could be useful.

4.5. Sensor Network Monitoring through Visual Analysis

In the following we present a visualization system for the real-time monitoring of sensor networks. This refers to Contributions $C_{4.5}$. One of the most prominent application scenarios is are supply networks such as gas, water or electricity. In the following we will focus on the latter.

In particular, it supports the control room operators of electric grids with large amounts of distributed power generation. It introduces the combination of an expert system with a visualization system, specifically designed for SmartGrid control rooms. The rule-based expert system filters the stream of a large amount of incoming events, searching for potential problems. As measurements are continuously read from connected Smart Meters, the expert system performs a classification of these events to relieve the operator from the manual inspection of irrelevant and trivial information items. As a result, the user is provided with the required situational awareness.

A coordinated, multiple view environment displays the electric grid model which is annotated with status indicators. This supports operators in the efficient monitoring of electric grids with a focus on distributed, small-scale power generation. Being a critical infrastructure, the electric grid is highly sensitive and any modification must be well justified. Our visualization system therefore also provides insight into the expert system enabling the user to validate and verify the expert system's analysis process. This provides the required decision support to assist the operator in keeping the grid in stable operating condition.

The analysis results are presented in a details-on-demand manner. This gives the operator both a reasonable overview on the entire system, but also enables him to inspect specific parts in detail. Other sources of information such as ICT coverage and weather conditions are included to provide additional context. The visualization system thus provides monitoring and control support for operators to keep the grid in a stable condition.

4.5.1. Combining Automatic and Manual Analysis

The task of the control room operators is to monitor events in the monitored networks. Based on their knowledge and years of experience, domain experts decide whether an event or a series of events indicates a problem in the grid or not. Where necessary, they must interfere and instruct counter-measures before the problem propagates through the grid causing cascading effects.

The visualization system we present supports the control room operators in their task of monitoring a large and complex system. It uses an expert system to analyze the measurement data and

displays all relevant information in a multiple view environment. A large overview visualization displays the current state of the grid and allows the user to get additional details on particular points of interest on demand. Not only the electric grid, but also other sources of information such as tangential infrastructure and weather forecasts are shown. Additional supporting views display the data from different perspectives. With their help, the operator can then gather related information to gain the situation awareness which is required for decision making.

In the electric engineering domain, an event is most commonly referred to as an alarm even if its content has purely informational character. Different automatic tools have been proposed to assist the operators. Most of them are about alarm processing, i.e. the pre-processing, filtering and prioritization [KSH07] of events. However, they typically operate in a "black box" manner, prohibiting insight into the internal workings. Thus, the user is not able to verify the correctness and is bound to rely rather blindly on the assumption that the reduced data set contains all vital information. Typical approaches include expert systems that are based on rule inference to analyze the system in a fully automated manner. The user has to rely on the fact that the a-priori knowledge encoded in the rules is sufficient. On the other hand, performing a manual analysis of all sensor readings is often hardly feasible, because it is difficult to have an eye on all sensors at the same time.

In this section we present a system that combines the strengths of an automated analysis with a visualization that enables the operator to track this process. We therefore define a knowledge base which contains a collection of so-called *facts* that represent all known entities and properties of the domain model. With a set of *rules* that describe how these facts relate, the inference engine can then derive new facts from existing ones. The combination of knowledge base and inference engine is commonly known as *expert system*. We use such a system to be able to automatically process large amounts of sensor data in real time. The strict separation of business logic and visualization also allows us to adjust and modify the domain-specific content without touching the view component. Even switching to an entirely different domain is possible with this approach.

The concept of Visual Analytics is about combining manual and automatic analysis in an iterative and alternating cycle which is an ideal choice for our analysis process flow. We think that it is crucial that no action is performed in fully automatic manner; the operator must have the final word on every decision.

Our expert system follows the pipeline design and is split into several distinct analysis steps. Suspicious events are first filtered and grouped according to the similarity of different intrinsic properties such as location or severity. Then, diagnoses over these events are put up before user recommendations are generated. The system keeps track on items that are linked across the pipeline stages so that the visualization can display which facts and rules led to the generation of which diagnosis and recommendation. With this setup, the control room operator can verify the correctness of how these assumptions were made and decide which action should be taken.

Our visualization system as a whole goes beyond pure monitoring: a dedicated view displays the analysis pipeline of the expert system that operates in the background. In contrast, our system enables the operator to trace the workflow pipeline of the expert system and thus verify its correctness. We are convinced that no action should be taken automatically – the operator must be the final decision maker.

4.5.2. Domain Model and Event Processing

In the following chapter we outline the concept of a monitoring system that combines the knowledge of an expert system with a visualization that conveys the gathered information from the sensor measurement network in a verifiable manner to the control room (see Figure 4.7). Based on that information, the operator can then control the actual model which influences future measurements.

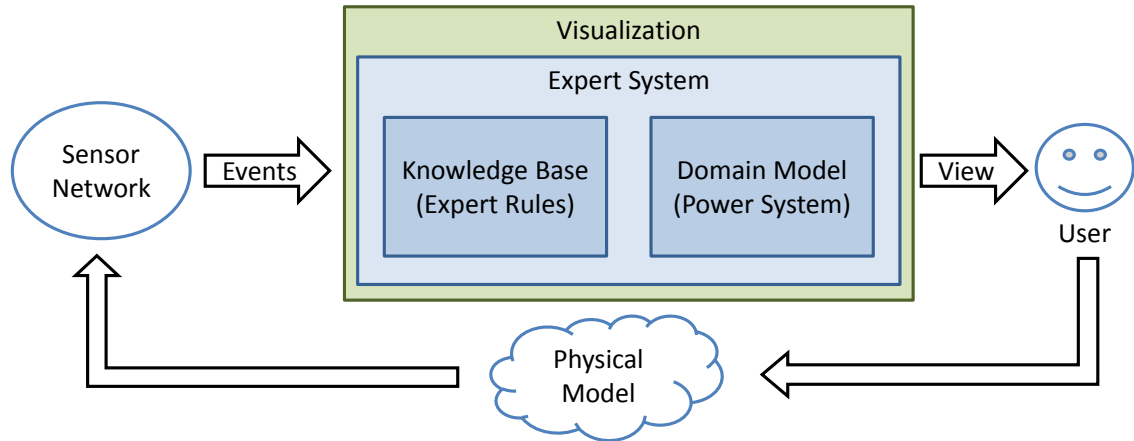


Figure 4.7: Incoming events are delegated to the monitoring system that internally uses an expert system to process the data. With the help of the visualization, the user applies changes to the physical model which in turn influences the measurements.

The foundation of the visualization system we present is generic, but we will discuss a customized version for distribution networks. Its underlying domain model contains transformer stations, distribution bars, meters, lines and the like. With respect to the ongoing changes in the way electricity is produced, we also integrated different of kinds power plants such as solar plants or wind turbines.

Today, most of the data in sensor networks is transmitted over the wire, but wireless solutions are gaining more and more traction in the community [EKM11, PKS10]. Cellular base stations are required to receive the sensor data from the system and send commands to the control elements. However, they require power which they receive from the electric grid they control. Ergo, failures in one system can cause the other system to fail as well. This hazard can be mitigated by ensuring that every grid element in the grid is covered by at least two base stations. Consequently, the coverage of ICT infrastructure with respect to Smart Metering and its controls plays an important role and needs to be represented in the model as well.

Measurement events arrive at high rates from the Smart Meters in the grid. The system processes and updates the state of the domain model accordingly and in real-time. Expert systems have proven to provide valuable support for this task in different application domains in the past. We use such an expert system to perform automatic assessment and filtering of incoming events.

Meter	Message	Result
19947 - SF VB	Out of Bounds	245000 NOT in range [220.0 - 240.0]
19177 - SF J03	Out of Bounds	245 NOT in range [220.0 - 240.0]
19947 - SF VB	Out of Bounds	243714 NOT in range [220.0 - 240.0]
19129 - SF J95	Out of Bounds	244000 NOT in range [220.0 - 240.0]
19809 - SF J01	Out of Bounds	244 NOT in range [220.0 - 240.0]
19129 - SF J95	Out of Bounds	244357 NOT in range [220.0 - 240.0]
19947 - SF VB	Out of Bounds	244000 NOT in range [220.0 - 240.0]
19486 - SF VB	Out of Bounds	242 NOT in range [220.0 - 240.0]

Figure 4.8: The list of problems as detected by the expert system in tabular form. This type of display is familiar to many domain experts and preserves the temporal order of events, but lacks spatial context.

Based on a set of user-defined rules, the system evaluates the measurement readings, filters them and generates diagnoses and recommendation actions where necessary. Most operators are familiar with the tabular display of system events. We transferred this metaphor to our system, so that operators would recognize known elements from conventional displays (see Figure 4.8).

4.5.3. A Rule-based Analysis Pipeline

The knowledge base provides the application context for the system. In our setup, it contains facts from the electric engineering domain such as transformer stations, overhead lines, voltage meters, measurement readings, etc. The collection of rules has been constructed with the knowledge of control room operators and defines how these elements relate with each other.

The expert system is designed as a pipeline with three distinct analysis steps. First, the large collection of alarms is filtered so that only suspicious items proceed to the next stage. All other events are discarded as early as possible which keeps the computational effort and memory requirements to a minimum. In the next step, the set of suspicious alarms is diagnosed and grouped by similarity leading to an even smaller set of diagnosis entries in the fact base. The information, which alarm contributed to which diagnosis is retained for the visualization. The third stage of the pipeline renders zero, one or more recommendations for each diagnosis. Again, the link between diagnosis and recommended action is recorded and displayed later on. See Figure 4.9 for an overview and an exemplary path through the expert system pipeline.

The expert knowledge in the system is used for the first, low-level analysis of the data and can be performed up to this point automatically – no user interaction is required. The generated set of recommendations can then be analyzed on a higher level by the user who can verify the correctness of the assumptions and conclusions that have been made. It is therefore essential to provide assistance for the human reasoning process, especially decision support to enable the user to decide which recommendation to follow.

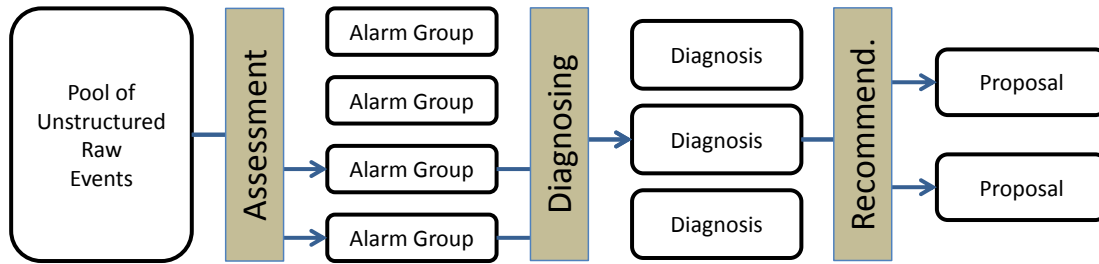


Figure 4.9: Pipeline configuration of the expert system. The analysis path leads from the pool of alarms to two groups of similar, suspicious alarms. These alarms can be explained by one diagnosis which is linked to two alternative corrective actions.

The results of the automatic analysis process are delegated to the actual visualization which conveys the extracted information to the operator. Visual domain elements are annotated with problem indicators and links to corrective actions where necessary.

4.5.4. Visualization & Interaction

The task of the visualization component of the system is to provide this decision support. The need for analysis in real-time and the provision of situational awareness – i.e. the context which is used to comprehend the meaning of status variables – impose additional requirements for the monitoring environment. The operator must be able to get an overview on the health status of the entire system quickly. We approach this challenge with a view environment that contains a set of coordinated views that display the data from different perspectives.

One important source of information for the visualization system is the fact based that is maintained by the expert system. Its structure is an analysis pipeline with the stages *filtering*, *diagnosis* and *recommendation*. Operators can use the visualization to trace the path that different input events take through this pipeline (see Figure 4.10 for an example).

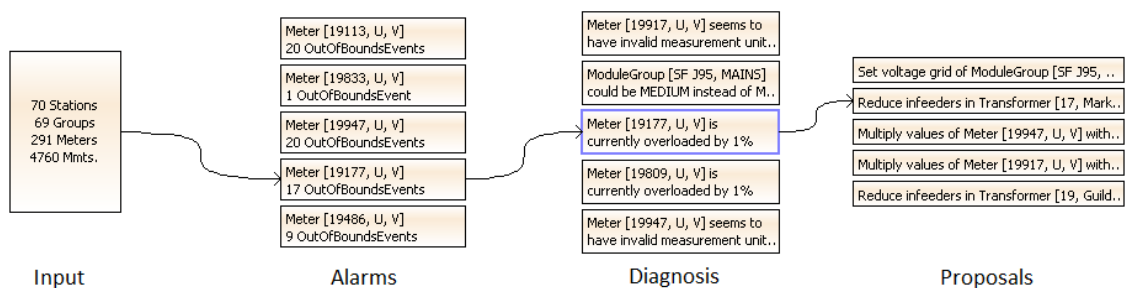


Figure 4.10: Input events are filtered so that only those which require attention are further processed. Based on that, diagnoses that explain the problems are put up. The expert system then recommends one or more curative actions.

The first main contribution, the display of the overall status of the system is displayed in a large view (Figure 4.16, top). Most importantly, the operator needs to grasp the "big picture", i.e. to get a comprehensive overview over the state of the entire system (see Figure 4.11). This is accomplished with a high-level display that uses geographic references to provide a spatial orientation for the operator, but displays domain entities only. A geographic map is not displayed underneath as it is typically not needed by control room operators. The only exception to this rule is the display of lines: overhead cables are more sensitive to disruptions than buried cables and are thus displayed differently. Transformer stations are displayed as rectangular elements that comprise its distribution bars and the installed meters. Initially, the view contains connected outlined rectangles that represent transformer stations at their respective geographic coordinates.

The color indicates the overall state of the station as determined by the expert system. Green stands for *tested and o.k.*, gray for *undetermined* and red indicates problems. Zooming in on a station "opens" it, giving more details on its distribution bars. Similarly, the font color of the voltage grid label indicates the status of all connected meters. One level deeper, individual meters are displayed. At the deepest level, the meters expand in size and display the latest measurement readings in a small linechart. Hovering over a defective meter with the cursor shows a tooltip that displays the latest alarm that relates to that meter.

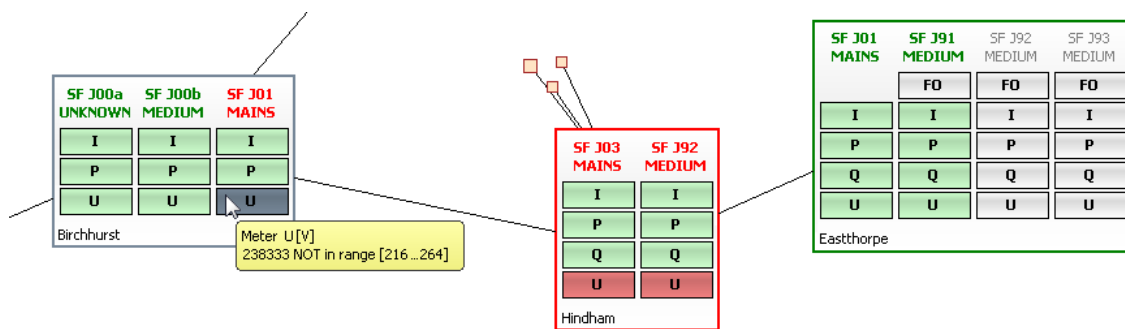


Figure 4.11: Detail of the overview visualization with three stations with several distribution bars each (columns). The frame color indicates the overall status while tooltips give details in text form on individual meters. Three solar plants are connected to the station at the center.

Energy suppliers are displayed on demand to indicate the sources of additional input. Their size and shape are illustrated by their respective glyphs. Lines to the connected transformer stations are also displayed to show where the electricity comes into the system. Whenever problems occur, the respective visual elements are highlighted with red color to indicate that the operator's attention is required. Additional details on a particular problem are available on-demand using tooltips.

To complete the picture, we also integrated the coverage of ICT infrastructure in the overview. On demand, communication towers are included in the overview display as glyphs. Their coverage area is represented as filled circles to indicate which station covers which transformer stations. Failures at a communication tower render the circle translucent. The color of the coverage areas is accumulative – overlapping areas become darker, uncovered areas become white. This helps the operator to find out for which areas a single failure would lead to a disconnection. As can be

seen in the example setup in Figure 4.12, the blackout at the communication tower can be traced to a failure at the transformer station nearby. Operators can use this display to create a mental link between these different infrastructures.

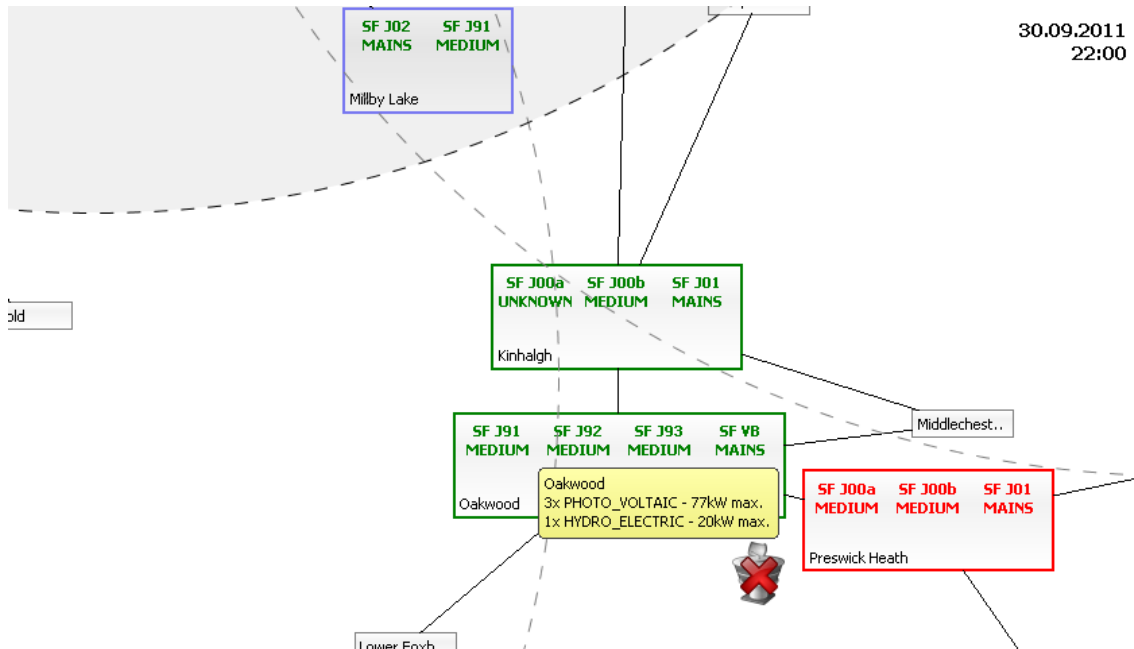


Figure 4.12: The overview visualization showing a blackout at one communication tower. The expert system reports problems at the station "Preswick Heath" which are likely to relate to the failure. The communication tower in the north west still covers "Milby Lake", but the other stations are cut off.

A major challenge here is to preserve the spatial and temporal context so that the operator can investigate an incident. We therefore display additional views to enable the operator to analyze the data model from different perspectives. If we were to depict all elements of the entire domain model on the screen at the same time, overplotting would occur. A common approach to deal with this is to use aggregation (i.e. group similar elements of a hierarchy) and to use a virtual camera (often referred to as *zooming and panning*). Depending on the camera's current zoom level, we group meters, distribution bars and transformer stations (see Figure 4.13). This enables the operator to quickly get an idea of the overall status. The zooming metaphor is employed to enable the operator to "drill down" where necessary. We also use the panning mode of the virtual camera to navigate in the Smart Grid domain model.

However, when aggregation is used, special care must be taken in order not to hide important information. Therefore, problem indicators propagate their state upwards to higher level elements if necessary. For example, if a meter reports overload but is not visible in the current setting, the transformer station that contains the meter is marked. This indicates that at least one element in the group requires the operator's attention.

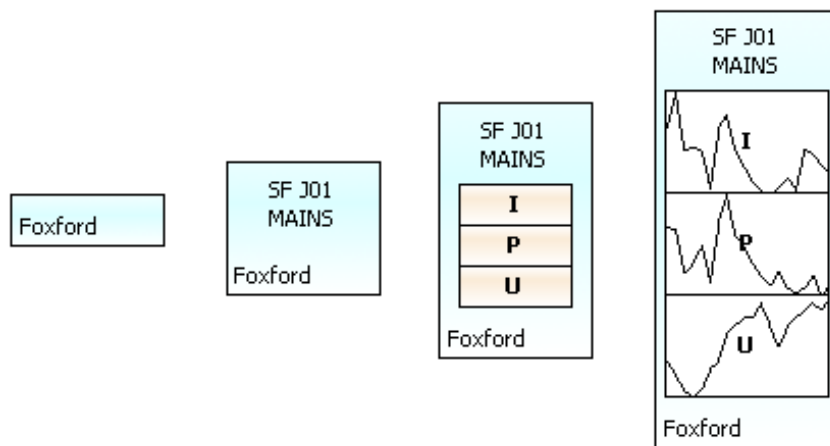


Figure 4.13: Zooming in increases the level of detail of the display. Fewer stations can be shown at the same time without overlap, but more information on a particular station of interest becomes visible.

More information on particular elements can be requested on-demand by hovering. A tooltip displays text messages according to the type and the status of an element. This information comes again from the expert system's evaluation process. Thus, also problem diagnoses and proposed fixes can be displayed and applied in-situ.

The user can interactively select a set of elements such as voltage meters in one view which causes the visualization system to select the same items in other views, too. This allows, for example, showing a meter's measurement readings in one view and displaying the geographic location of its transformer station in a different view. This coupling metaphor is typically referred to as *brushing & linking* [Kei00] and is used throughout the system.

Selecting an element indicates to the system that the user is interested in more details. It then automatically updates all linked views that can display information about this element. In Figure 4.14, three views display the status of a voltage meter with respect to current and historic values. The historic data is aggregated on a daily and yearly basis. Two line charts display the values as box plots where each group represents 50% and 99% respectively. They are linked with the live data view through time: the boxes that receive the current measurement values are marked. The temporal context enables the operator to decide whether the current measurements are typical for the current month and weekday.

Equally important is to display current weather conditions for the region. The majority of the small energy providers is highly dependent on either solar radiation or wind. Thus, knowing the current conditions enables the operator to derive how much energy is produced at these sites. Even if the amount of produced electricity is not for the privately owned plants, the overall power consumption of the connected transformer station can be analyzed. In Figure 4.15, the solar radiation of a sunny day is put in juxtaposition with the recorded power consumption at a particular secondary substation. The average power consumption for that day for central Europe is displayed as an overlay to allow for quick comparison between expected and actual values. Together, they

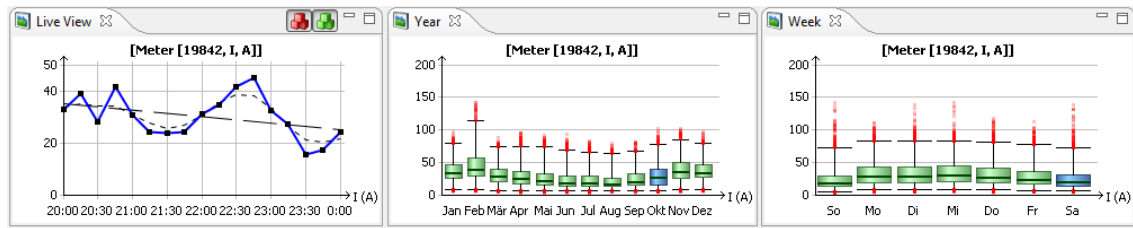


Figure 4.14: Three linked analysis views that put a meter's measurement readings into context. The Live View on the left shows the actual values together with frequently-used support functions such as Moving Average or Linear Trend. The other two views display historic values aggregated into box-plots. Every box represents 50% of the data value within a certain period of time (here: month and weekday). The whiskers enclose a total of 99% of the data, the remaining 1% is displayed as individual dots to indicate outliers. In each view, one box is marked with blueish color to indicate its correlation to the values in the Live View. Thus, the operator can derive whether the latest measurement readings are typical with respect to the month and the weekday.

make the strong correlation between solar radiation and the reduction in power consumption for that station apparent.

Displaying the analysis process of the expert system is the second major task of the visualization. The analysis view displays the expert system pipeline. Starting on the left side with the pool of events, suspicious elements are transferred to the second stage. In Figure 4.16 (bottom), the alarms are grouped by the meter they are related to. Selecting a group opens it and displays all contained items, selecting a single alarm highlights the linked domain entity in the overview display. The third column displays all diagnoses and the last one shows the set of recommended actions.

The view draws connections from one stage of the pipeline to the next on demand only, because the total number of links can be quite large which makes them difficult to follow. Thus, the view allows for the interactive tracking of the path of specific alarms through the analysis pipeline. The two views are linked, i.e. selecting an element also selects it in the overview, thus providing a spatial context. Where necessary, the camera aligns the overview so that the selected meter becomes visible. Similarly, selecting a recommendation in the analysis view displays a preview of its effect in the overview. This can be, for example, the display of throttling values for energy providers.

4.5.5. Discussion

In this section we outlined a monitoring system that combines the real-time processing capabilities of an expert system with an operator-friendly multiple view environment. It supports the monitoring and controlling of electric grids with a focus on distributed generation. Data from connected Smart Meters at the transformer stations can be viewed and analyzed. In contrast to conventional systems, it also integrates other sources of information such as weather conditions and ICT coverage.

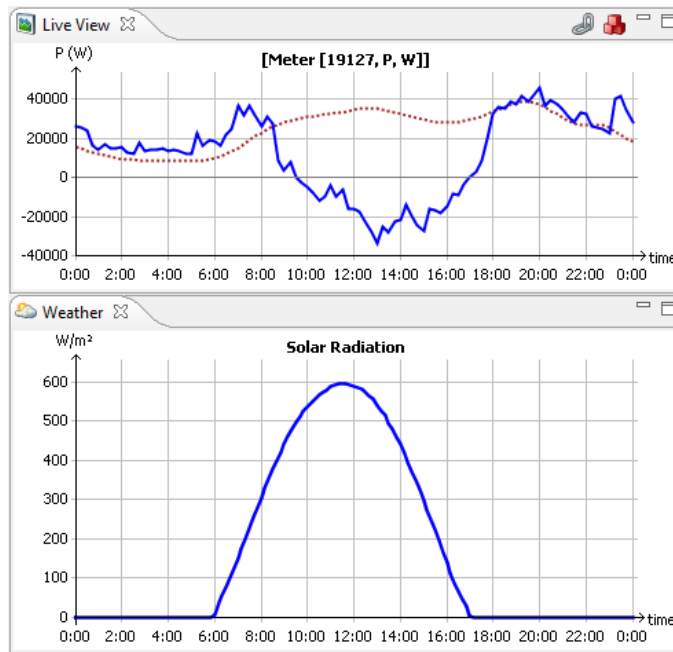


Figure 4.15: *Top: power consumption of the selected station versus average power consumption in central Europe (dotted line). Bottom: Measured solar radiation at the closest weather station. The correlation with between the two can be explained by the high amount of connected solar plants. In fact, the power consumption goes significantly below zero indicating a surplus up to 20kW for sunny days.*

Using an expert system makes it possible to quickly assess incoming measurement events and to update the displayed domain elements accordingly. This expert system is designed to work as an analysis pipeline and its workflow is displayed in the visualization. This enables the operator to track the processing path of alarms and verify the correctness of the system. An overview visualization complements this approach by providing a temporal and spatial context to the discovered issues. First indicators show that the combination of expert system and visualization can provide valuable decision support for operators. The integration of renewable energy providers together with weather data has been implemented successfully, but an exhaustive description is beyond the scope of this thesis. This additional information aims to improve the situational awareness of the operators, but must be integrated in the system so that the user can easily understand them.

Supporting the operator to create a virtual model of real-life conditions without the need of a computer scientist would simplify the knowledge engineering process significantly. Going one step further, evaluation of planned new lines and stations in a virtual environment with simulation support can bring substantial benefits for planners. The simulation of the effect that different action alternatives have on the system is an important aspect is discussed in Section 4.8.

As a whole, the system provides situation awareness for control room operators, but also enables them to get diagnosis information and choose corrective actions.

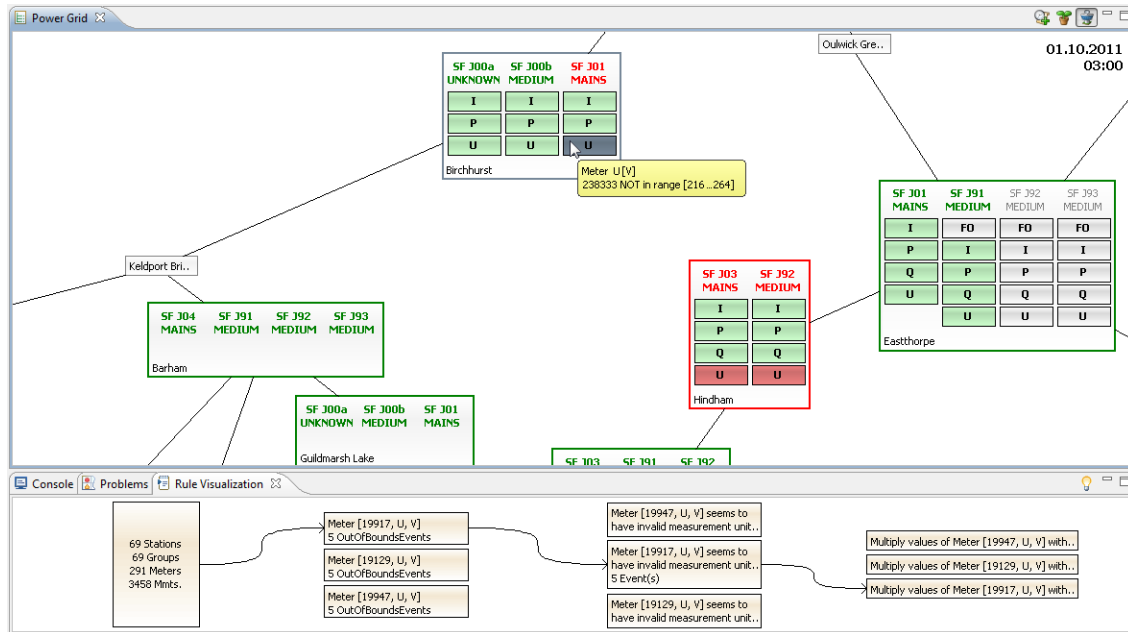


Figure 4.16: Screenshot of the running system with its different views. In the overview (top), transformer stations are displayed together with all contained meters as rectangles, grouped by distribution bar (mains and medium voltage grids). The background color of the individual meters reflects the analysis result of the expert system, (green=ok, gray=unknown, red=problem). Tooltips give details on discovered problems. The status is propagated upwards in the hierarchy causing the frame of a station to become thick red if any contained meters report problems. Selected items are tinted with blueish color. In the analysis view (bottom) the user can get a detailed explanation on why a certain meter has reported problems and what the expert system recommends to cure this.

4.6. Anomaly Detection in Sensor Networks

In this section, we extend the system to support exploratory tasks for the comparison of univariate, geo-referenced sensor data, in particular for anomaly detection (Contribution C₆). We split the recordings into fixed-length patterns and show them in order to compare them over time and space using two linked views. Apart from geo-based comparison across sensors we also support different temporal patterns to discover seasonal effects, anomalies and periodicities.

The methods we use are best practices in the information visualization domain. They cover the daily, the weekly and seasonal and patterns of the data. Daily patterns can be analyzed in a clustering-based view, weekly patterns in a calendar-based view and seasonal patterns in a projection-based view. The connectivity of the sensors can be analyzed through a dedicated topological network view. We assist the domain expert with interaction techniques to make the results understandable. As a result, the user can identify and analyze erroneous and suspicious

measurements in the network. A case study with a domain expert verified the usefulness of our approach.

4.6.1. Revealing Similarity of Time-series Data

Typical approaches include expert systems based on rule inference to analyze the system in a fully automated manner. These software systems usually operate in a black box manner that do not allow for user interaction. The user has to rely on the fact that the a-priori knowledge encoded in the rules is sufficient. We refer to Section 4.5 for an in-depth discussion of the issue.

Performing a manual analysis of all sensor readings on the other hand is often hardly feasible, because it is difficult to have an eye on all sensors at the same time. Also, it is quite difficult to derive trends and frequently occurring patterns from simple line chart plots. A set of visual tools can successfully assist the human in the analysis process to become more effective and efficient.

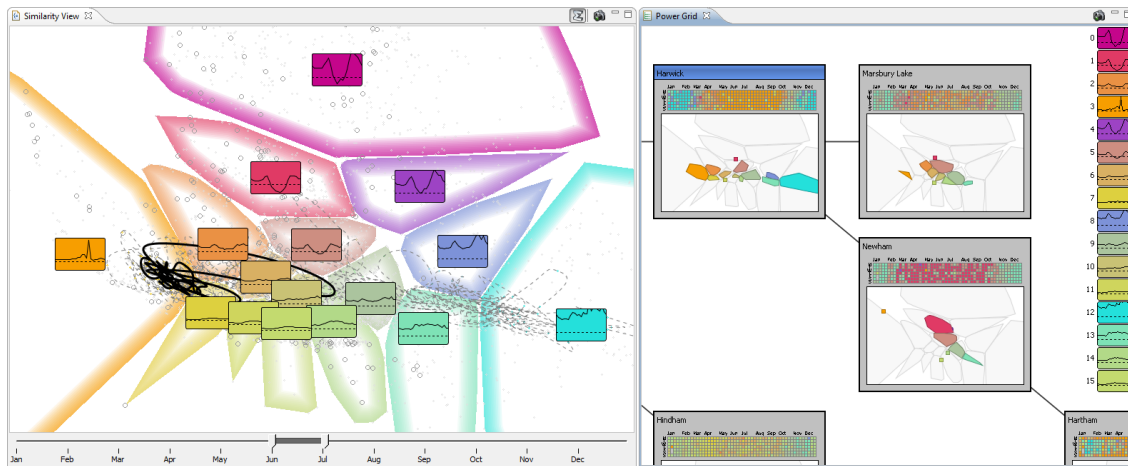


Figure 4.17: A screenshot of the entire system. The similarity view on the left side shows all daily patterns of all sensors. Similar patterns are assigned to the same group and color. The change of patterns over time for a selected sensor is indicated by the black spline. The network view on the right side gives an overview of the network topology. The small calendar in the node glyph shows changes over time and a fingerprint view underneath shows the sensor patterns in the global context.

The users that work with this kind of data are, for example, operators in the control room of an electrical grid. Their task is to monitor the state of the network using the measurements provided by the installed power meters across the country. The operators must be able to identify repeating patterns as well as anomalies with respect to changes over time and across different sensors. For that, it is important to recognize diverging states, anomalies and suspicious patterns as quickly as possible. More specifically, we have identified the following problems:

- Getting an overview of a sensor network is required in order to get an impression of the “big picture” and to identify potential problems.

- The user needs to analyze the network in space and time to find atypical patterns in the network.
- Comparison of different sensors at the same time but also the development of a single sensor over time is relevant.
- Based on the pattern similarity, the user must be enabled to quickly identify non-standard patterns and trends.

Based on these tasks, we derive a set of design criteria. Atomic entities (i.e. daily patterns) are analyzed with respect to three different criteria: based on content and relations to the geographic and temporal context. These different aspects must be linked to enable the analyst to provide additional insight and to solve multi-criteria problems. We identify the most appropriate visualizations with respect to the properties of the data and the user task:

- The data is recorded at several linked univariate sensors that measure the same physical quantity.
- The time-series data can be segmented into meaningful equally-sized day-long patterns.
- Interesting patterns are expected to be daily, weekly or yearly.
- The system must be able to robustly detect and deal with outliers and missing values.

We contribute a visualization system that is able to assist the analyst in dealing with these problems. It consists of two tightly coupled views that complement each other: a *Similarity View* and a *Network View* (see Figure 4.17). A topological map of the network gives a geo-based topological overview on the network in space and development of patterns over time for every sensor. Using a calendar-based visualization, the analyst is able to identify trends on different scales, based on individual sensors. As a result, the user can identify erroneous and suspicious measurements in the network. A similarity-based view gives important details on the global relations of different temporal patterns (in our example the power consumption over the day). The user can thus analyze daily patterns of the sensors, grouped by their pair-wise similarity. On demand, points that belong to the same sensor can be connected. This gives the analyst a quick overview on the variability of daily patterns over a period of time. If the patterns are very similar this spline would look like a tiny hairball and anomalies can be easily spotted. Tight linking between the two ensures that the user recognizes the same element and sets of similar elements in both views.

4.6.2. Data and Algorithms of the System

In this chapter we present the data and algorithms for the visualization system. We do not use pattern shapes directly but provide similarity-based measures to support the identification of similar and different patterns, as well as projection and clustering techniques. In this way we support the user in the identification of both frequent patterns as well as outlier patterns. We first cover the data preprocessing routines before we explain the rationale behind the design decisions. Before

we start, the data is condensed from high-dimensional data vectors to lower-dimensional feature vectors that are presumed to contain the majority of information and are faster and more robust to work with. We compute the similarity based on this data, before we aggregate similar patterns into groups. This allows the user to get an overview of recorded measurements and identify trends and repeating patterns in the visualization.

4.6.2.1. Input Data

The input data is a collection of time series measurements of a single variable spanning over one year. The sensors that record the measurements are organized in a network structure. Nodes indicate sensors, the edges between two nodes indicate some kind of connectivity.

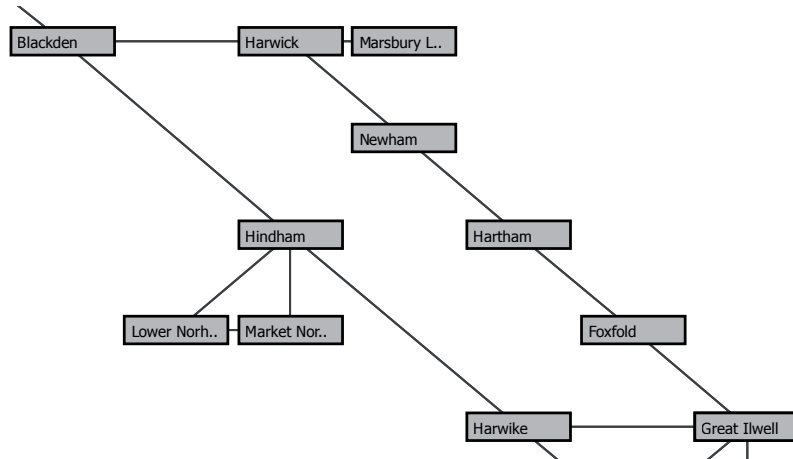


Figure 4.18: A part of the sensor network, displayed as a octi-linear topological map. At the cost of uniform edge lengths, geographical directions are preserved, if possible [NW06].

Not all sensors have recordings for every time stamp, and day, respectively. Since missing values may be an important aspect for analytical tasks, our visual representations need to be sensitive to an explicitly defined missing value indicator. Another aspect of the data is that some patterns are partly filled with zeros. In some scenarios, zero can be interpreted as missing value, whereas in others, it cannot. We therefore do not assign a special meaning to this value. We prefer a shape-preserving in favor of a domain-preserving pattern comparison strategy. Thus, we apply a *standard score* normalization for the input data per sensor. To reduce the impact of outliers we previously apply a moving average procedure with a kernel range of one hour. The next step in the analysis process is the segmentation of the time series data into individual patterns. In our scenario, diurnal variations are the smallest repeating patterns and therefore the segmentation into days appears to be the right level of granularity.

4.6.2.2. Similarity Measures

Many different algorithms for measuring the similarity of time series data exist. Our system supports different analysis tasks and therefore supplies different similarity measures. For the

analysis of values or changes in the values, the Euclidean distance is a useful measure to compare patterns. We argue in accordance to Hadlak et al. [HSCW13] that trend-based similarity measures support the user in finding simultaneous changes over time well. If the shape of the consumption pattern is of interest, the correlation coefficient and Dynamic Time Warping (DTW) are reasonable choices. The DTW algorithm compares two time series by aligning sequences of the data so that the distance between the two is minimal [BC94]. This makes DTW robust to shifts and length of the temporal sequences. While the original version is rather expensive to compute – it is in complexity class $O(n^2)$ – several speed improvements have been implemented since then. We use the optimized FastDTW algorithm as described by Salvador and Chan [SC07].

4.6.2.3. Dimensionality Reduction

In order to make large data sets accessible to the user, a variety of data reduction techniques exist. One of the most used projections is Principal Component Analysis (PCA). Being a linear method, it is very sensitive to outliers and does not use the available display space too well. Multi-Dimensional Scaling (MDS), a group of methods for dimensionality reduction, is also very popular. Since its original presentation [Tor52], many variations have been developed [Kru64] and has gained popularity also in the graph drawing community [BP09]. Also, non-deterministic projection methods such as Stochastic Neighbor Embedding have been employed [HR02]. Using locally restricted projections, Joia et al. present not only a new projection approach, but also a comparison of different approaches [JPC*11]. Another survey is presented by Lee et al. who discuss dimensionality reduction schemes without user supervision [LV10].

4.6.2.4. Dimensionality Reduction Quality

With the reduction of data comes a loss of data quality. Many different measures are available that assess the quality of a given projection. A natural choice, in particular for MDS methods, is to use the weighted stress function as an indicator for the projection quality. Kruskal proposed a small variation of this stress function as well as some reference values for quality [Kru64]. However, measuring the quality with the same means as the actual algorithm seems to be an unreliable choice for MDS. Sips presents “...two quantitative measures of class consistency, one based on the distance to the class’s center of gravity, and another based on the entropies of the spatial distributions of classes...” [SNLH09], which are robust against outliers. In the work of Bertini et al., an overview on many different quality measures, pros and cons as well as application domains can be found [BTK11].

4.6.2.5. Projecting Similarity

The generated distance information is rather extensive and not directly interpretable by the analyst. At this point, dimensionality reduction becomes necessary to be able to convey the information to the user. The user needs to be enabled to detect changes, especially outlier patterns and to find clusters of similar patterns.

Inspired by projection-based approaches such as the MotionExplorer system [BWK*13], we derive a 2D projection of the time series patterns based on the pair-wise distances. The goal here

is to preserve the distances from the original data set as good as possible. Patterns that are similar should have 2D positions that are close and patterns that are very different should have a large distance between them. Here, the first part of the statement is more important than the second one. If two very similar patterns are plotted apart, the user gets a wrong impression of the data. On the other side, if two different patterns are far apart, it is not that important how different they are. This allows us to use non-linear projection methods that preserve local structure in favor of global projection quality.

The resulting scatter-plot represents the similarity of the daily patterns. Any kind of projection introduces errors, due to the expected loss of information. After a series of tests, the class consistency measure of Sips et al. turned out to be the most robust quality measure [SNLH09]. For each point, the set of n nearest neighbors in high-dimensional space is compared to the n nearest neighbors in 2D space. The quality is defined by the set of elements that appear in both sets. We use this approach to assure that the projection quality is high enough to allow for drawing reliable conclusions from the data. In practice, stress-based non-linear projection methods such as those from the MDS family perform quite well for many data sets [JPC*11].

4.6.3. Visualization & Interaction

We present two tightly coupled views of the linked time-series data to the domain expert. Based on an atomic entity – i.e. daily patterns – all data records are arranged based on their pair-wise similarity and displayed in the Similarity View (Figure 4.17, left). This provides an overview of the recorded patterns in the network and allows for different interactions. In a second, geo-based view (Figure 4.17, right), the same patterns can be analyzed on different levels of granularity in time, but also in space using a visual calendar. The visual link in between is based on the color that is assigned to the different groups of daily patterns. In combination, the two views show the data from two complementary perspectives, both supporting each other.

4.6.3.1. Similarity View

The Similarity View gives an overview on all recorded patterns of the entire network. It displays the patterns with respect to their pairwise similarities that were computed in Section 4.6.2.5. Every pattern is represented by a single point in the screen space.

By selecting a particular node in the network, the analyst can investigate the change of patterns over time. The view connects all daily patterns of that stations and orders them by time. The result can be seen in Figure 4.19. All days of the station *Newham* are plotted for the month May. The patterns oscillate at a high frequency up and down with one outlier on the left. In contrast to straight line segments, bézier splines are used to interpolate between the patterns, because the changes are expected to happen gradually.

Using a range-based slider that spans over the entire year, the user can filter the data set with respect to recording time. Filtered elements are not being hidden to preserve the context, but they are rendered small and their color becomes faint. The filtered part of the spline turns into a thin, gray, dashed line. See Figure 4.19 for an example.

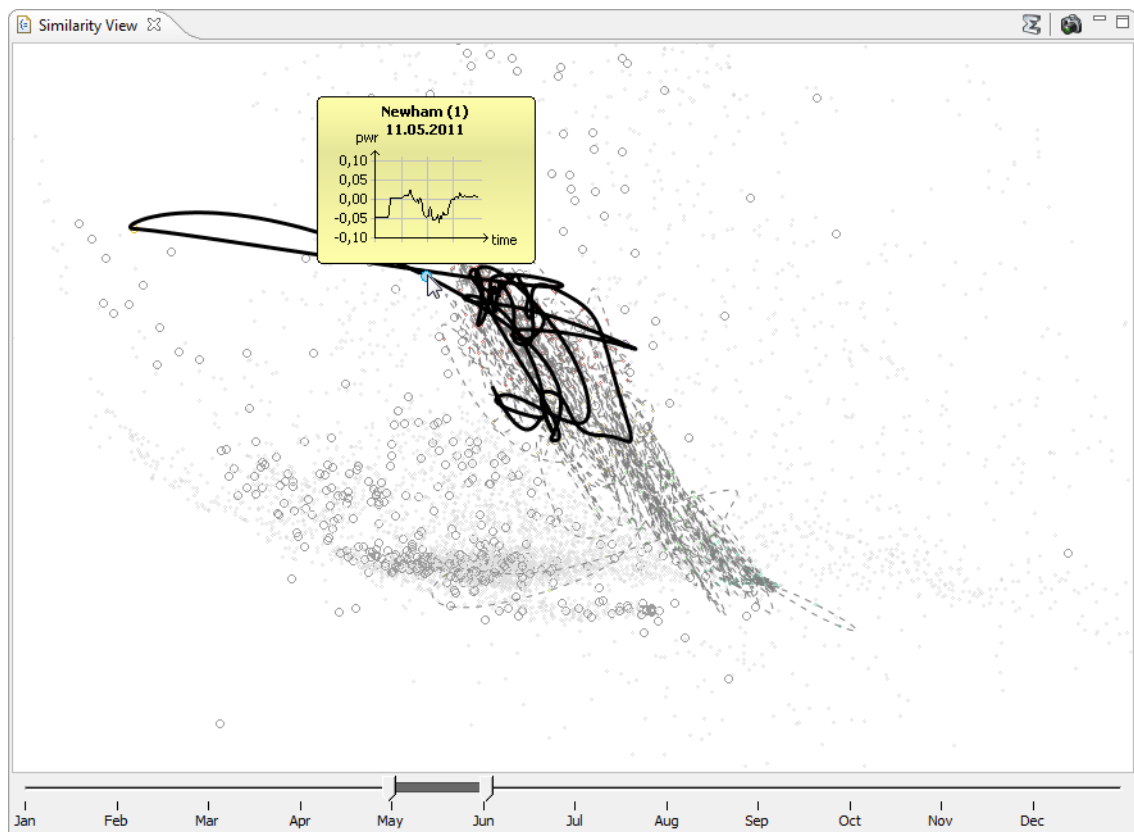


Figure 4.19: The sensor at Newham, plotted based on the similarity of daily patterns for the month May. The thick line represents the month May as indicated by the time slider at the bottom. The gray dashed line represents the rest of the year.

The user can access the actual shape of the pattern on demand by hovering over any element. A tooltip with additional information about the data point appears, providing details about the sensor, the recording date and a line-chart of the time series of the particular day.

As already mentioned, the axes in this display do not have an intrinsic meaning. The information in the scatter-plot is very fine-grain and does not permit an intuitive understanding of its organization. We create an abstraction layer on top of the scatter-plot display. In other words, we discretize the continuous space into a set of discrete partitions. In this layer, similar patterns are aggregated into larger groups which help the user in getting an overview of available patterns and their location. This is accomplished by clustering of the data and displaying the groups, each annotated with a representative element. Clustering is performed in 2D space, after projection of the data, to prevent clusters from being rendered as fragmented regions. The result of this operation can be seen Figure 4.20.

Generally speaking, the intrinsic property of good clusters is that the elements within have low pair-wise distances while distances to elements in other clusters are comparatively large. If a

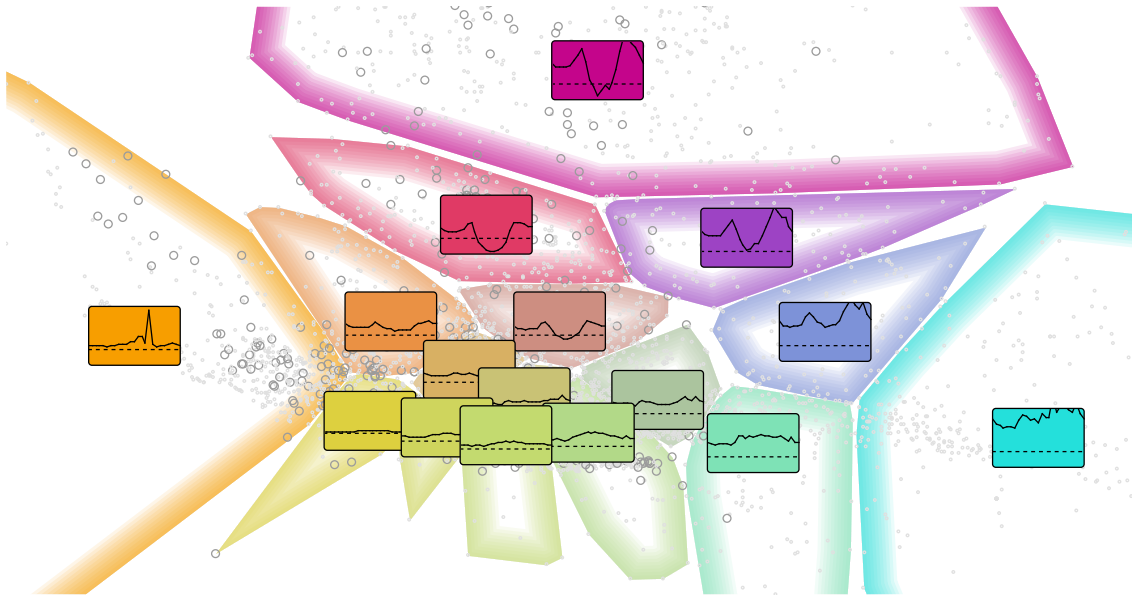


Figure 4.20: *The projected pattern similarities, clustered and colored using a 2D color map. Some of the patterns show negative energy consumption during daytime (purplish red patterns at the top). This could indicate that connected solar plants produced significant amounts of energy on that days.*

central element in the cluster can be identified, it would represent the other elements in the cluster with a minimum of lost information. The k-means algorithm creates a clustering based on such cluster representatives. While this algorithm is rather basic, it produces the cluster representatives. This pattern stands for the means of the cluster, i.e. an artificial pattern with the smallest distance to all other pattern in the cluster. In cases where no new element can or should be created, the closely related k-medians algorithm works on existing elements only [JD88]. The common challenge of choosing an optimal number of desired clusters is not a problem in our main use case, because its main purpose is to create a simplified version of the data. The number of clusters is limited by the number of pattern shapes the user is expected to differentiate. In practice, choosing k in the range of 10 to 20 seems to be reasonable.

To improve the readability of the drawing, every cluster is annotated with a small line chart glyph of the mean pattern that the cluster stands for. This technique is similar to the Micro-Macro Views display [BvLBS09], which uses the rectangular grid of a SOM to derive the 2D position of the entities together with a representative for each of the clusters. In our system, the clustering is separate from the projection, resulting in a non-rectangular layout of the clusters. It is shown as small line chart in the center of every cluster.

We emphasize similar patterns using a discrete set of colors to indicate cluster membership. Thus, the color indicates the shape of the pattern without having to show the actual pattern. Patterns of similar color are expected to have a similar shape. Using the 2D position in the similarity plot, the corresponding color of a pattern can be derived from a 2D color map. This allows us to



Figure 4.21: A 2D color map created by interpolation of four perceptually distant colors. It defines the color of the similarity clusters.

also use the color as an indicator of similarity. However, using only one color per clusters makes it easier for the user to recognize a certain color as the same in another view if many different slight variations coexist. The color map must enable intuitive and accurate readings in order to express the metrics of similarity. On the one hand, it should exploit a maximum of different colors. On the other hand, the user must be able to estimate the approximate distance between two objects correctly, which requires a perceptual uniform interpolation. In contrast to the RGB or the HSV color space, CIELAB is a non-linear colorspace that can be used to extract perceptually uniform 2D planes. However, as presented by Bremm et al. [BvLBS11], these color maps do not contain many perceptually different colors. Inspired by the work of Ziegler et al. [ZNK07] we use four perceptually distant colors and interpolate between these colors. However, we slightly use a different set of colors, namely yellow, cyan, red and blue. The goal of this selection is to separate the color map into complementary color tones and also from fully saturated (bottom) to fully intense (top) colors. We use cyan instead of green in order to approximately equalize the perceptually distance between all corner colors. The corner colors are equalized in intensity and saturation in the HSI color space [Kei00] and then interpolated in the CIELab color space. The color map is depicted in Figure 4.21.

While this view already contains a lot of information on the occurring patterns in the network, the network structure is not visible. Also, it is not immediately clear, which sensor measures which pattern at which time of the year. We overcome these limitations with a second view that displays just that. Tight coupling and interactive linking between the two ensures that the user can bridge the mental gap between two different visual representations of the same entity.

4.6.3.2. Network View

This second part of our system has its focus on the network topology. The visualization is a node-link diagram with drill-down functionality that displays temporal information in the node glyphs on demand. In this manner, the user can not only learn about the spatial organization, but also the pattern distribution in different temporal granularities.

Nodes represent sensors and edges denote connections between the sensor. A sound layout should create an intuitive display of the topology, but preserve directions, if possible. The user is interested in an abstraction of local geographic coordinates to reduce the visual complexity of the network. General graph layout algorithms, however, try to satisfy edge length constraints and/or minimize the number of edge crossings. This are typically not problems for sensor networks, as both criteria are not overly important.

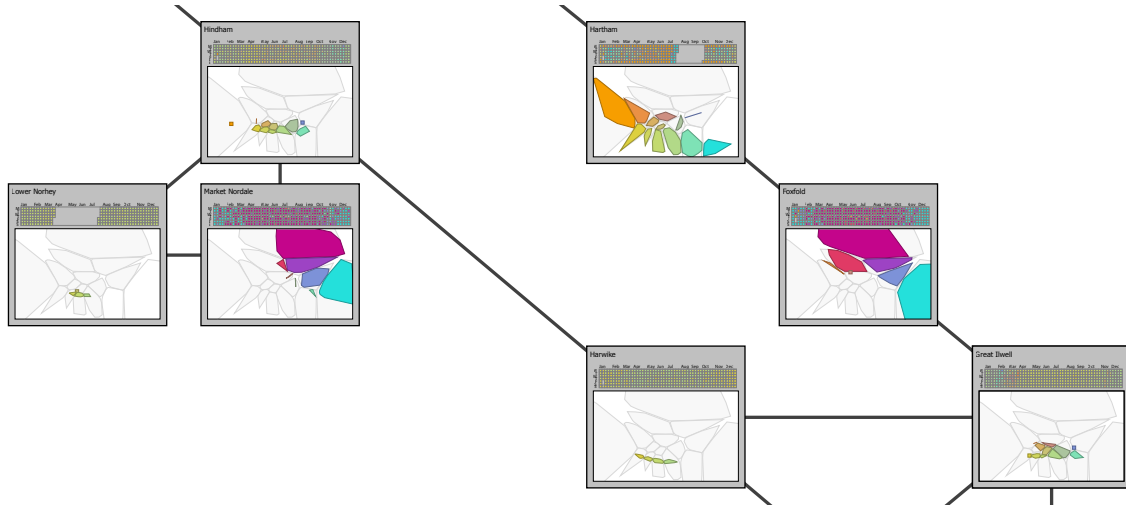


Figure 4.22: *The network view at the second level of detail. Both calendar and cluster fingerprint view appear.*

A prominent group of methods that achieves this is the octi-linear layout family. These algorithms create a schematic representation that is inspired by the metro map metaphor. Originally, these methods were used to generate layouts of subway lines, which lead to the name *Metro Maps*. They restrict the angles of edges between nodes of the network to multiples of 45 degrees, yielding a stratified version of the original layout. They also try to preserve directions where possible. We adopt one of these algorithms to compute the layout of a sensor network. While different algorithms exist, we chose the work of Nöllenburg et al. [NW06]. In contrast to other works, it favors quality over computation speed. As the layout is static, this can be pre-computed and thus speed is not a major issue.

In order to avoid cognitive overload, this view uses different level of details to adjust the visual complexity. A “virtual camera” that supports zooming and panning enables the user to navigate in this spatial view. Zooming in and out triggers different levels of detail of the sensor node glyphs. At the most abstract level, all nodes are represented by simple, labeled rectangles (Figure 4.18). Starting at the second level, higher zoom levels show two small views: the calender and the cluster fingerprint (Figure 4.22). Every zoom level scales the views in intervals, because only discrete scale factors make sense for the contained calender view.

Focusing on a particular station, the user is interested in its behavior over time. The patterns in the contained time series can be analyzed with respect to different occurrence frequencies. Using

van Wijk's Calendar View [VWVS99], we assign a single color value per day based on the cluster the pattern belongs to. This color is then used to colorize a calendar (Figure 4.23). In this manner, the user can thus identify seasonal patterns. In contrast to radial plots, the calendar view assigns the same amount of screen space to individual patterns, which gives them equal visual importance.

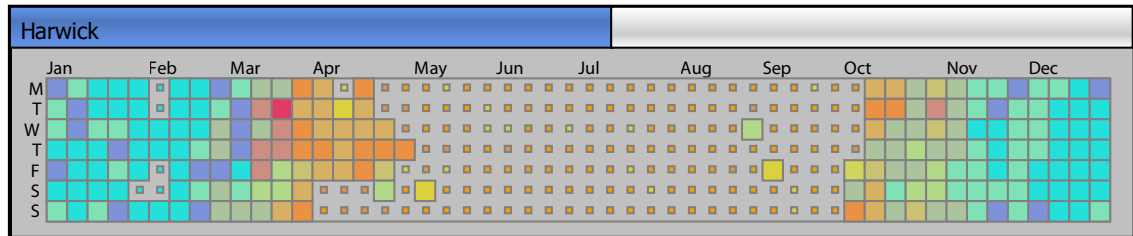


Figure 4.23: Calendar view of a partly selected sensor. The calendar maps colored patterns to a cluster. A selection is active which causes unselected elements (mostly in summer) to become smaller.

Every day in the calendar is colored by the cluster color this day belongs to. While different layouts for calendars exist, we decided to align weekdays on along horizontal axis. Weekdays are ordered according to the international standard ISO 8601 which defines Monday as the first day of the week. This alignment brings Saturday and Sunday together, which facilitates the distinction between workdays and weekends. From left to right, weekly patterns and changes over the year for a given weekday become apparent. From top to bottom, patterns within a week are visible. Looking at a distance on the small calendar, larger seasonal changes are most recognizable. A tooltip shows the actual pattern together with the date and the ID of the cluster. Using cluster IDs serves as an alternative to matching the color across different views, in particular for people with color deficiency.

The analyst also wants to know which patterns are specific to a particular sensor. We therefore added a small filtered version of the similarity view. All clusters are displayed in light gray to provide context to the current focus (the sensor). Then, a filtered set of clusters that contain only patterns from this sensor is created. In a Focus & Context approach, these reduced clusters are then drawn on top of the faint, unfiltered clusters. Patterns that were recorded by the sensor in focus are highlighted using the same set of colors. This creates a visual fingerprint of the sensor that also has its representation in the similarity view. As in the similarity view, the clusters are drawn using their convex hull, similar to the work of Schreck and Panse [SP07]. Again, tooltips indicate the cluster ID to differentiate borderline cases. See Figure 4.24 for an example.

The system also shows a legend on the right side of the view to facilitate the matching between pattern and color. It is based on the representative pattern of the cluster and the color that is derived from its location in the color map. It enables the user to see which color relates to which pattern. Again, corresponding IDs are displayed to differentiate borderline cases. The displayed glyph contains the representative pattern of the cluster which is also used in the similarity view. This strengthens the link between the two views.

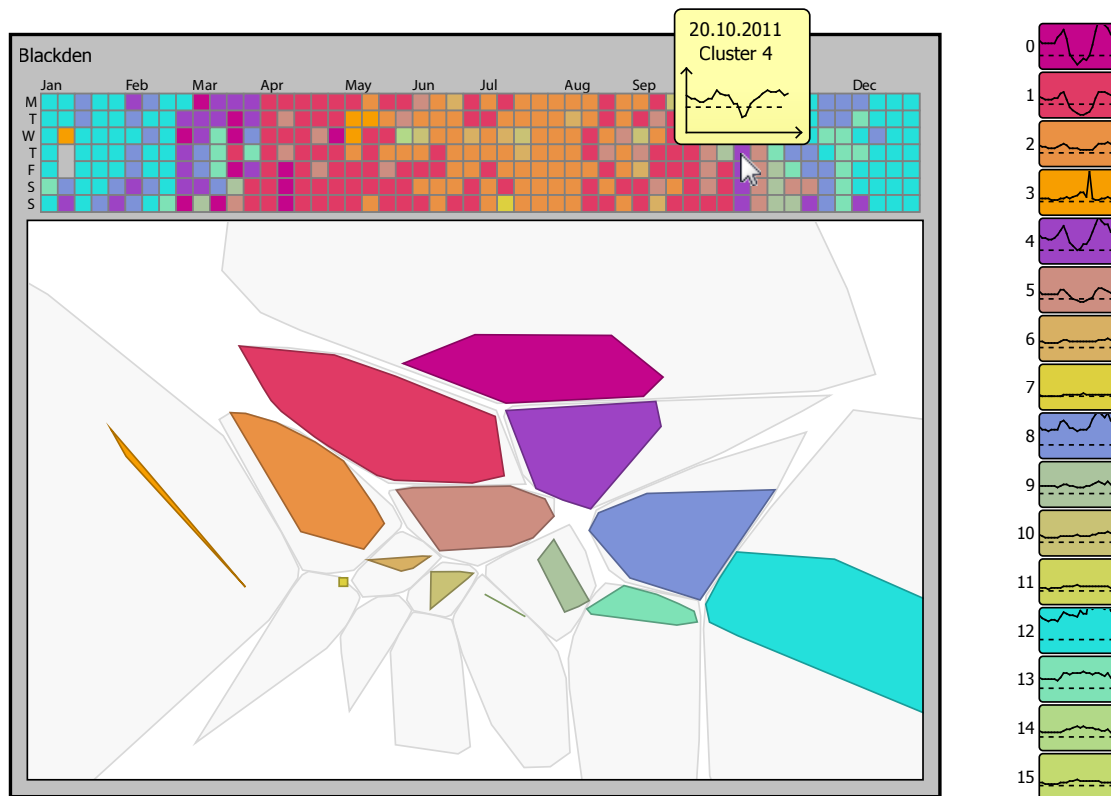


Figure 4.24: A sensor shown at the highest level of detail. The calendar maps time to a cluster of patterns. The fingerprint view below illustrates which patterns this sensor recorded compared to the other sensors. Low and even negative consumption patterns are recorded from March to October.

4.6.3.3. Linking the Two Views

Aside from the visual linking between the two views, interaction with one of them can also affect the other. Selecting a sensor in the network view triggers the selection of all linked time series patterns in the similarity view. Using a single selection color to highlight a selected element would overwrite the cluster association of the elements. We therefore use the color of the corresponding cluster to highlight selected patterns and display the remaining ones in gray.

On the opposite side, we can also select interesting patterns in the similarity view and see their distribution in the network. We use a lasso selection tool that is known from image manipulation software to maximize flexibility. Again, selected patterns are colored while unselected patterns remain gray.

On the lowest level of detail, the network view shows the distribution of selected patterns across the network. We use a progress bar metaphor (blue bar on bright background) to reflect the fraction of patterns that were selected. As can be seen in Figure 4.25, the selection affects mostly the sensors at *Hartham* and *Harwick*. About two thirds of the sensor at *Hartham* are selected.

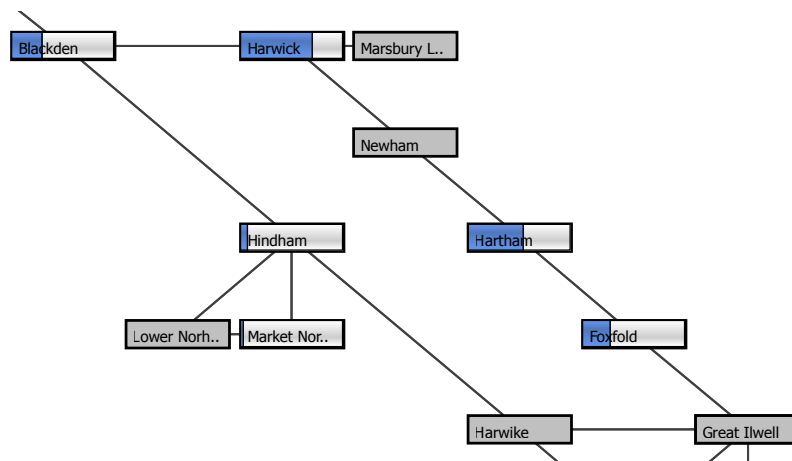


Figure 4.25: Selected patterns distributed to their related sensors in the network. Sensors that are at least partly selected are accordingly marked with bluish selection bars on bright background.

The analyst can also zoom in to also see the selection distributed to individual days. As can be seen in Figure 4.26, mostly Saturdays and Sundays are selected. In this example, only patterns on the left part of view have been selected. Thus, selected patterns are in different variations of orange. The cyan clusters are not part of the selection and do not appear in the calendar view. Filtered patterns are drawn as miniaturized rectangles to indicate that they are not part of the selection. Missing values are not drawn. The fingerprint view is not affected by the selection.

4.6.4. Case Study

We performed a guided case study with a domain expert to demonstrate the usability of our approach in a real-world use case. The expert identified two major areas of relevance: monitoring and planning. The first step was to identify interesting patterns with the help of the legend of the network view (Figure 4.24). In the legend, the pattern that occurred most frequently gave the expert a quick overview on the network. An interesting finding was the prominence of patterns with backflow (i.e. patterns with significant values below zero) during daytime which is unusual. These patterns indicate an electric flow from the consumers back into the grid – an often undesired result which is due to the high amount of solar panels in the pilot region where the data was recorded.

In the next step, the grid was explored using the Network View. The domain expert first focused on the calendar view, because it was considered the most intuitive one and most similar to the tools the expert uses. Typically, manual lookup of patterns from the previous years is required to derive typical daily patterns, based on the day of week, season of year and other circumstances (e.g. public holidays). For the monitoring task, the focus was on some of the previously identified patterns (see Figure 4.24). After that, the Similarity View was used to select the interesting parts (patterns with back-flow) in the top-left corner using the lasso tool. This selection action highlighted in the Network View that most of the patterns were recorded at only 5-6

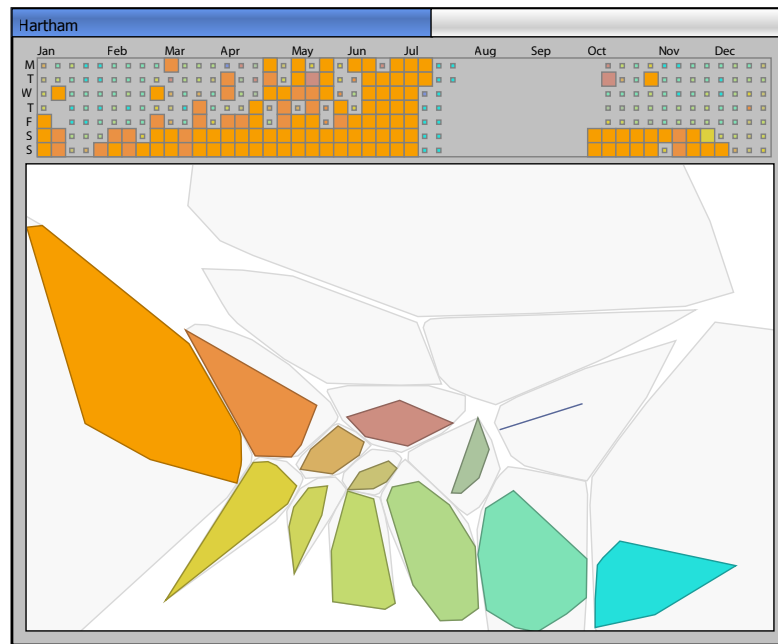


Figure 4.26: Partly selected sensor at Hartham. Mostly weekends are affected by the selection. Filtered patterns are displayed only as small rectangles. The months August and September do not contain any data.

stations in the network (e.g. in Figure 4.25). The expert concluded that only these few stations needed to be investigated further in terms of backflow protection. For the planning task, the interest was on finding the right time to temporarily isolate stations or cables for maintenance. This should be done when power flow is at the lowest for all relevant stations. The expert therefore used the network overview to anticipate the pattern for different station on a given day based on the recordings of the previous year.

4.6.5. Design Process

In order to optimize the design choices, we performed the design process in an iterative manner. Different data mappings, visual representations and interactions were explained to a group of 8 non-expert users and two experts from the electrical grid domain. We conducted informal interviews with the running prototype which led to fruitful discussions about the pros and cons of different aspects of the system. In a final round, we gave a video demonstration to two usability professionals to get feedback on the usability of the system.

The first idea was to create a geo-referenced layout that is drawn on top of a thematic or navigational map. A result from the interview with the experts was that geographic reference is required only in exceptional cases. The most important design factors for them were the network topology, followed by simplicity.

The visual representation of the sensor node has changed significantly through the design phase. One idea was to split the calendar into four distinct seasons. The fingerprint view was motivated by the fact that users could not correlate the two views without explanation. Putting the calendar above the fingerprint was motivated by the fact that it was unclear to some of the users where the calendar legend belongs to if put the other way round.

Different concepts to connect the similarity trajectory of a single sensor were proposed. The idea of drawing arrow heads to indicate the direction of the spline was rejected, because the glyphs were often misinterpreted in crowded displays. Aside from line segments and splines, convex hulls [SP07] and bubble sets [CPC09] were evaluated. While they two emphasize areas, they also cover much screen space, especially when outliers are present. Also, the temporal sequence was no longer visible. Combining multiple techniques seemed promising at first, but produced too much overplotting. We conducted a survey with about 15 non-experts with 12 screenshots of the system, each with a different color map. It clearly confirmed that the four colors we used achieved the clearest color separation.

Using integer IDs for clusters was suggested by one of the users to enforce the ability to recognize the same cluster in different representations, especially for color deficient people. The ID is used in the legend and in the tooltips of the calendar view, the fingerprint view and the similarity view.

4.6.6. Discussion

In this section we presented a visualization system for interactive pattern analysis in univariate sensor networks. The focus is on the analysis of similar patterns over different temporal scales, but it also respects the network structure of the sensors. It consists of two strongly linked views that enable the analyst to gain insight into the data set. The cluster prototypes show typical, often occurring patterns. The network view gives an overview over the network topology and the patterns for each sensor. This enables the analyst to compare different sensors and to see seasonal trends.

We considered two types of scalability: the number of stations and the length of measurement data. The application is fairly robust with respect to the number of nodes. The similarity view is not affected by the network complexity and the network view uses a drill-down metaphor to adjust the amount to displayed information. For very large networks, aggregation based on either topology or geography could be used. Currently, only one year of measurements can be analyzed. Comparing yearly patterns requires a different visual encoding of the data.

4.7. Explorative Analysis of 2D Color Maps

Color is one of the most important visual variables in information visualization. As discussed in Section 4.6, one prominent example is the encoding of similarity in perceived similarity of colors. In many cases, two-dimensional information can be color-coded based on a 2D color map. A variety of color maps as well as a number of quality criteria for the use of color have been presented. The choice of the best color map depends on the analytical task users intend to perform

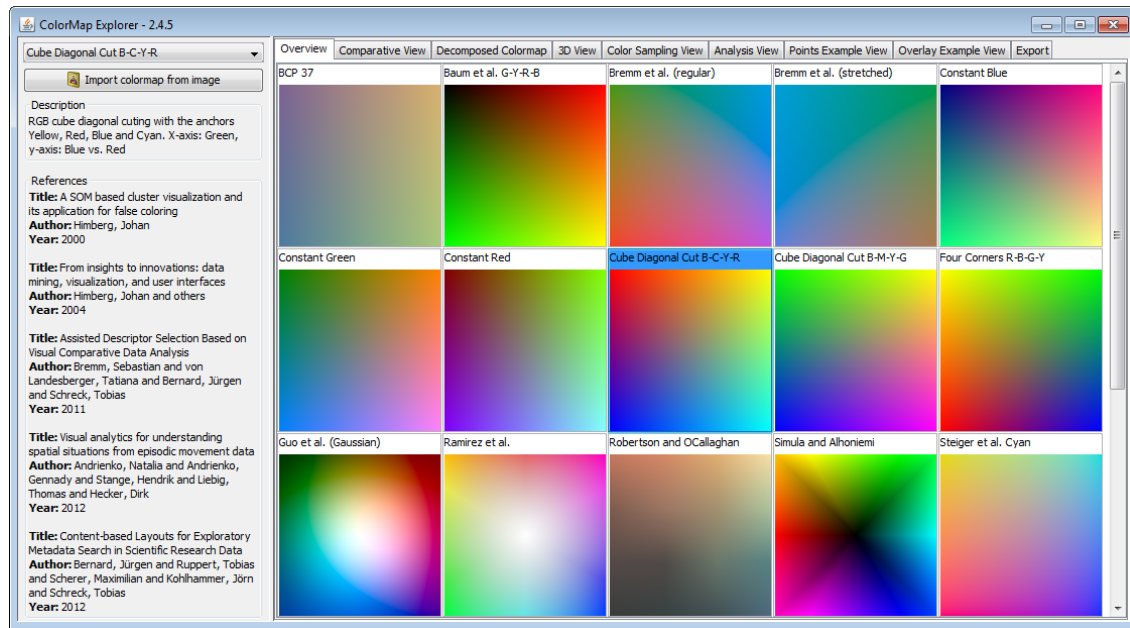


Figure 4.27: The main window of the ColorMap-Explorer: the config and info panel is placed on left side, the collection of views is stored in individual tabs at the right. The Overview tab enumerates all available color map implementations.

and the design space in choosing an appropriate 2D color map is large. In this section, we present the ColorMap-Explorer, a visual-interactive system that helps users in selecting the most appropriate 2D color map for their particular use case (Contribution C₇). The ColorMap-Explorer also provides a library of many color map implementations that have been proposed in the scientific literature. To analyze their usefulness for different tasks, ColorMap-Explorer provides use case scenarios to allow users to obtain qualitative feedback. In addition, quantitative metrics are provided on a global (i.e. per color map) and local (i.e. per point) scale. ColorMap-Explorer enables users to explore the strengths and weaknesses of existing as well as user-provided color maps to find the best fit for their task. Any color map can be exported to be reused in other visualization tools.

4.7.1. Visual Access to Color Maps

Depending on the properties of the underlying data, different types of color maps can be applied to encode data attributes visually in the most accurate way. Qualitative color maps allow for the distinction between different categories of elements. Quantitative color maps allow for an identification of similar (and dissimilar) data elements with respect to a quantitative value domain. For quantitative color maps, the most relevant representatives are either sequential (unipolar) or diverging (bipolar). In those cases where a single data variable (attribute) is encoded, a one-dimensional color ramp can be used.

For high-dimensional data, 2D color maps are used to preserve similarity of the items in a visual variable. Data items with more than two attributes are first mapped into the two-dimensional space according to some transformation or projection method. The result of these upstream techniques is a mapping in 2D that can directly be used as position information in a 2D color map.

As a result, the viewer can estimate the relative similarity of high-dimensional data by comparing colors. As such, 2D color maps are appropriate for high-dimensional data; we do not recommend the direct use of two data attributes as coordinates in the map (cf. Wainer et al. [WF80]).

A variety of different static 2D color maps has been presented in the past. The survey of Bernard et al. gives an overview [BSM*15]. The authors review quality criteria and design guidelines for color maps and depict the huge design space for the *design* and the *use* of static 2D color maps.

In order to faithfully reflect the relative pair-wise distances of the original data as closely as possible, such a 2D color map should preserve the notion of perceived similarity in terms of color. The perceived distance between colors should be linearly related to the geometric distance in both the high- and the 2-dimensional space. Another quality criterion for a color map is to exploit the given color space, aiming for a maximum number of distinguishable colors. In many cases the choice of color maps is also made with respect to colorblindness sensitivity. For example, about 8-10 percent of the male population in Europe suffer from a color vision deficiency [Alb10]. Additional requirements to color maps may be based on user-centered constraints like corporate designs. In some cases, 2D color maps may also require a certain contrast against the background color so that the visual elements can be clearly identified as such. Some other visualizations may require that text and other overlays are legible on a canvas that is drawn based on the color map.

A taxonomy of different color map design criteria is presented by Tominski et al. [TFS08]. According to the authors, meaningful color encodings strongly depend on the data, the task, the target user group, and the display device. A fourth dimension in the problem space is the large number of static 2D color maps presented in literature. Naturally, there is no color map that is perfect with respect to all requirements. To give an example, a color map can hardly be colorblind-safe and maximize color exploitation at the same time. Visualization designers need to balance a trade-off between different complementary design criteria. A premature color map decision may lead to false assumptions with respect to the underlying data properties. Consequently, choosing a 2D color map for a visualization should be done carefully.

To the best of our knowledge, a decision support system that supports the user in making such a choice has not yet been presented. We identify the following challenges:

- Visual overview of existing color maps
- Comparison of color maps with respect to global quantitative quality measures
- Assessment of local properties of an individual map
- Visual analysis of the shape of a color map with respect to different color spaces.
- Assessment of the maximum amount of discernible information that can be encoded in a color map
- Showing the homogeneity of perceived similarity

- Assessment of the interplay of color map with other visual variables

We present the ColorMap-Explorer, a visual-interactive decision support system for 2D color maps. The system assists visualization designers to find the best-fitting color map in this complex search space. At the moment, it contains 22 color map implementations that were discussed in the scientific literature. Visual access to these 2D color maps is provided in an overview visualization. For every color map, quantitative metrics are provided on a global (i.e. per color map) and local (i.e. per point) scale. For the comparison of multiple color maps, we provide a view utilizing the global measures. A detailed analysis of local properties is provided by several views, each shedding light from a different perspective. In particular, we allow for the detailed analysis of a) different color channels b) local perceptual linearity, and c) the shape of the area in different color spaces for every color map. In order to get a first impression of how the color map behaves in a targeted downstream visualization, several views stress the color map against other visual variables in different example scenarios. Finally, the selected color map can be exported for re-use in downstream visualization tools.

The workflow of the the ColorMap-Explorer is as follows: starting with an overview of all color map implementations, the user can select up to three color maps which then are put in juxtaposition. This allows for direct comparison to narrow down the number of candidates with respect to the analytical task. Individual color maps are then investigated in more detail before the best fit is identified. When the decision on the best matching color map has been made, the user can save the color map as an image to disk. The user can always move backwards and forwards in this workflow pipeline as desired.

4.7.2. Encoding Information in Color Maps

Appropriate color maps for specific tasks and specific data properties is a well discussed topic in the literature. General guidelines on selecting color maps can be found in [RO86, War88, RTB96, Rhe00]. In addition, linear color ranges (1D) for segmentation and categorical data have been discussed previously [Hea96, HB03a]. For two-dimensional color maps there are few guidelines available. The study of Wainer et al. [WF80] showed that encoding of two dimensional data with two dimensional color maps is not intelligible. In contrast to this statement, Ware and Beatty [WB88] found that each additional color dimension (red, green, blue channel) is as effective as an additional spatial dimension in the encoding of multivariate (more than two dimensional) data. As described in [MBS*14], there is a difference of between encoding single data dimensions with color and encoding (multidimensional) data relations. The first case requires a precise mapping of one data dimension to one color dimension or a one dimensional color map. The second case involves multiple dimensions for each visual object whose characteristics and relations to other objects should be revealed by color.

In [MBS*14] the authors present data-driven quality measures that are used to perceptually optimize color mapping for high-dimensional data. These measures are very effective if and only if the data set and its distribution as well as subsets (e.g., classes or clusters within the data) are known apriori and should be preserved in the color mapping. In this section, we focus on a data-independent approach, which focuses rather on the analysis tasks and not on data properties.

In multivariate data analysis applications, two dimensional color maps have been successfully applied [RO86, Him98, SA99, GGMZ05, GCML06, ZNK07, SvLB10, BvLBS11, AAS*12, BRS*12b, BWK*13, GP13, SBM*14, BSW*14] (see Figure 4.28 for an overview of implemented color maps). From this background, many two dimensional color maps have been proposed in the literature, each with different strengths and weaknesses. A recent survey has been conducted by Bernard et al. [BSM*15], enriched with a quality assessment for different tasks. Our work uses their quality metrics and provides them in an interactive manner to the user. In many aspects, our tool is similar to *PRAVDAColor*, an IBM software module that aims at supporting the user in selecting the right color map [BRT95]. Its main feature, however, is a set of perception-based rules that makes suggestions depending on task and data type.

4.7.3. Perceived Color Differences

For the rest of this section, we will refer to a measure that indicates how similar two colors are. Such a measure takes the human visual system into account in order to be reliable. In this section, we give a definition of a metric for measuring perceived color differences. This difference is 0 if two colors are perceived as equal and 1.0 if and only if the difference between two colors is “just noticeable” (visible by half the observers).

This definition of perceived color difference is based on the standardized color appearance model CieCAM02 [MFH*02]. Luo et al. have defined a distance measure ΔE for this model, based on the idea that a CAM should be a natural candidate to define a ΔE because similarity of colors should be rooted in their appearance attributes [LCL06]. The authors correlate appearance attribute differences to the color difference data sets and obtained a color difference formula and different parameterizations for the formula. They report that the predictive performance of the overarching CAM02-UCS parametrization is comparable to the specific parameterizations for small and large distances. This property is of particular importance for the evaluation of color maps, because it enables a quantification of the expected errors occurring when taking color distance for data distance.

A shortened definition is given in Equation 4.1, for a complete discussion, we refer to the original work of Luo et al [LCL06]. The definition is based on color appearance attributes J , M , and h (other choices turned out to have inferior predictive power with regard to color difference) and constants K_L , c_1 , and c_2 . The constants serve the purpose of fitting to small color distance (SCD) and/or large color distance (LCD) data, resulting in CAM02-SCD, CAM02-LCD respectively and CAM02-UCS (i.e. uniform) when fitted in combination.

$$\begin{aligned}
 J' &= \frac{(1 + 100c_1)J}{1 + c_1J} \\
 M' &= (1/c_2)\ln(1 + c_2M) \\
 a' &= M'\cos(h) \\
 b' &= M'\sin(h) \\
 \Delta E &= \Delta E_{UCS} = \sqrt{(\Delta J'/K_L)^2 + \Delta a'^2 + \Delta b'^2}
 \end{aligned} \tag{4.1}$$

Discounting for the constants, ΔE_{UCS} is an euclidean distance defined in a suitable derivate of CieCAM02. As an effect, J is being expanded by about 20%, with the coefficient c_1 actually being constant across the SCD, LCD, and UCS variants. On the other hand, the colorfulness M' , is being compressed significantly, with noticeable differences between CAM02-LCD and SCD variants. This hints at unexplained psycho-visual differences in the chroma component when judging small and large color differences. However, most color maps do not rely on chromatic content alone to differentiate colors. The hue h remains unchanged.

In summary, ΔE has the properties of a distance and a good approximation of perceived global and local color differences. Despite minor uncertainty regarding the role of the chroma component, it seems a very good assessment tool for quantifying the relation between value distance and perceived color distance inherent to color scales. Being based on CieCAM02, ΔE_{UCS} could even account for differences in lighting conditions and surroundings, but this has not been studied. The measure is thus based on standard lighting conditions, the sRGB “typical lighting conditions” representative for office use. It is therefore not an absolute measure; the actual perceived difference in color depends on many environmental factors such as lighting conditions, display, and the visual system of the user.

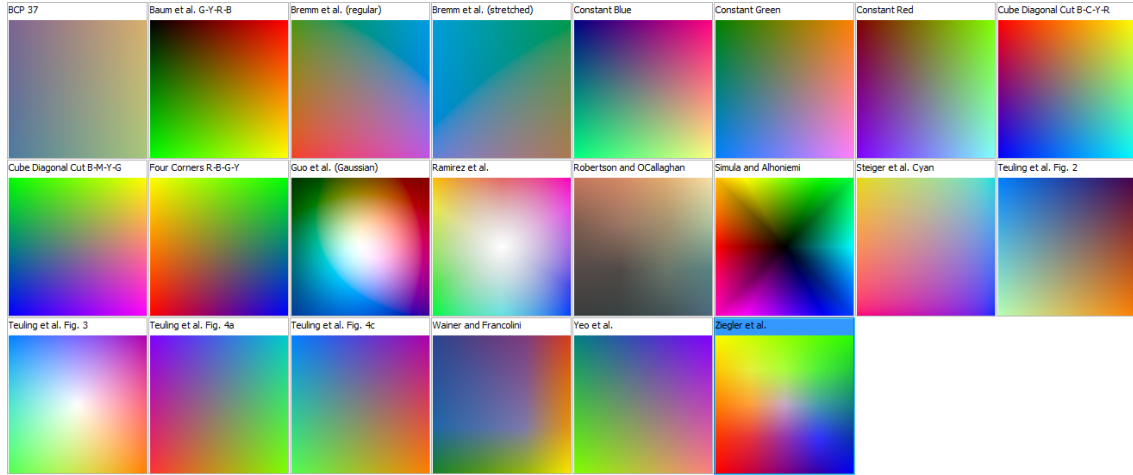


Figure 4.28: *The initial overview lists all available color map implementations.*

4.7.4. The ColorMap-Explorer

In this section, we present the different views of the ColorMap-Explorer along a typical workflow. The different views support the decision making process by showing individual features of the color maps. See Figure 4.27 for an overview screenshot of the software tool.

In total, 22 color maps mentioned in the information visualization literature have been re-implemented based on either functional description or digital images in publications. Software developers can extend the publicly available system by adding new color map implementations. In addition, an image-based file import enables non-experts to import custom color maps into the

system. Thus, visualization designers can easily extend the set of color maps and compare new designs with existing ones.

4.7.4.1. Overview Panel

Figure 4.28 shows all available color map implementations that currently exist in the ColorMap-Explorer. The Overview Panel lists all implementations as iconic images, annotated with name tags. The decision making workflow typically starts with this visualization, as this enables the analyst to gain an overview. In this juxtaposition, the visualization designer can narrow down the set of candidates to the most relevant ones.

Criteria for this filtering step may be based on user preference such as the existence or lack of specific colors. The display device is yet another restricting aspect. For example, foreground and background colors influence the applicability for the visualization design. In addition, the analytical task may be a limiting aspect for the set of relevant color maps. A guideline for the fitness of specific color maps with respect to specific analytical tasks has been discussed by Bernard et al. [BSM*15]. Other criteria could be based on color theory or perceptual aspects such as brightness levels.

Individual color maps can be selected to get additional meta information such as scientific publications that define or reference the color map. In these publications, the user can find additional information on the construction, usage scenarios, etc. (see Figure 4.27, left). This information can be used to further narrow down the collection of candidates.

4.7.4.2. Comparative View

The Comparative View allows for the direct comparison of the most relevant candidates. See Figure 4.29 for an illustration. For every color map, six complementing quality measures indicate the fitness for a given analysis task (cf. [BSM*15]). These quality measures assess the global quality of the color map with a single value; we therefore refer to them as *global* measures. For every quality measure, score, and ranking information is provided to facilitate the comparison with all other color maps of the system. A box-plot chart displays the mean score (red line mark) and the range of 25% and 75% quantile (pink background). The 10% and 90% quantiles are indicated by a thin line (the whiskers). The color maps and their quality measures are arranged on panels that are put in juxtaposition. By that means, the visualization designer is enabled to directly compare global quality aspects of different color maps.

As a result, the visualization designer can further reduce the number of candidates. An individual color maps can be analyzed further in the Decomposed View.

4.7.4.3. Decomposed View

The Decomposed View allows for the detailed analysis of single color maps. Two complementing aspects are considered. First, the color map is split into a set of color attributes from different color spaces. Second, local features of every attribute can be analyzed.

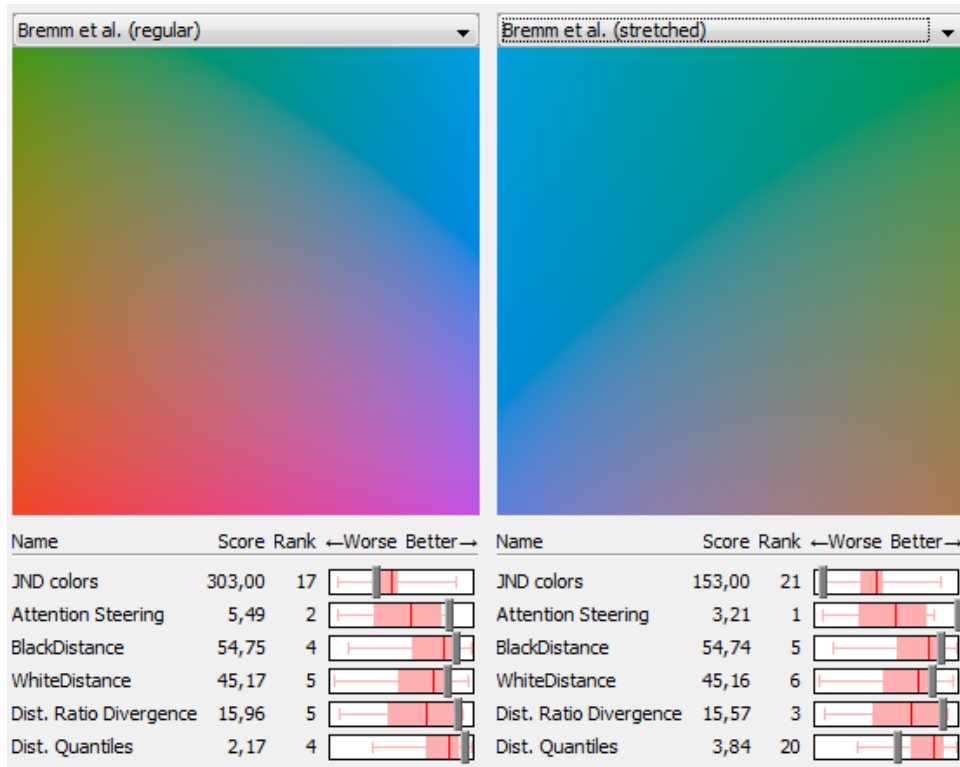


Figure 4.29: The Comparative View shows two selected color maps and their relative scores in different categories. This enables the user to compare scores and find the color map that fits best.

Viewing multiple color attributes In order to get an in-depth understanding of the properties of the color map, the map can be viewed from eight alternative perspectives. Each of them shows the same color map, but is filtered by a different color attribute.

We provide views for the red, green, and blue color components (center row in Figure 4.30), hue, saturation, and brightness (bottom row) as well as luma and attention steering (top row). Hue is special in that it highly depends on saturation. Without saturation, the value of hue is meaningless. Therefore, the tiles in the hue view are scaled according to its saturation. The original color map is shown in the top left panel. The first six values are directly extracted from the RGB and HSB color models.

Studies of Camgöz [CYG04] show that humans are predominantly attracted by bright and saturated colors. Attention steering effects may be harmful in several visual analysis tasks, because the analyst may be misled by striking features in the visualization that suppress less visual prominent features or patterns. Therefore, we approximate the potential of colors to attract the analyst's eye with $\sqrt{J^2 + M^2}$ where J is the relative lightness and M the colorfulness. This definition accords to the findings of Camgöz et al. [CYG04]. However, it is an approximation of the attention steering effects and is yet to be evaluated. Therefore, we show the both components J and M as decomposed views.

As a result, the homogeneity of a color map can be assessed. It also reveals how the color map is constructed. For example, the color map shown in Figure 4.30 is constructed by three diagonal color ramps in the RGB channels.

Revealing local characteristics The spatial distribution of color in the different filtered views yields a variety of local features that can be validated. We support the user in identifying *variations* across the map with glyph-based annotations. As can be seen in Figure 4.30, the display of the individual views is discretized. This allows us to enrich the view with local glyphs, similar to vector field arrow grids that are well-known in the SciVis community.

We chose regular hexagons as spatial discretization, because this reveals equal spatial distances between tiles and all neighbors (in contrast to rectangular tiles). The number of tiles is automatically adjusted according to the viewport dimensions. Thus, the user can adjust the discretization level.

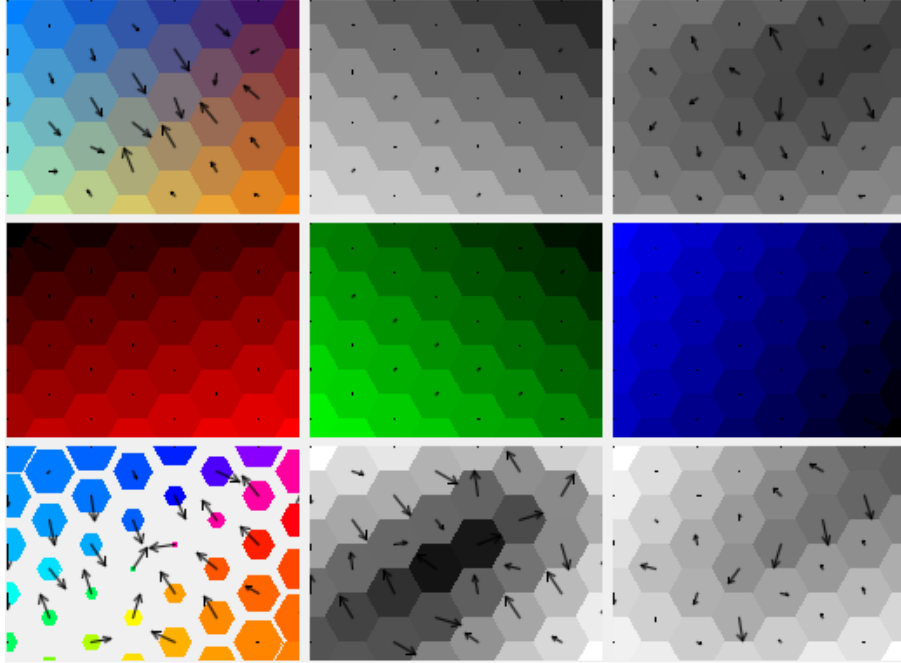


Figure 4.30: *The color map (top left) is split into different components such as the color channels. The arrows point in the direction of the strongest perceived color change.*

By default, each tile is annotated with a black arrow that indicates the perceived color distance with respect to its neighboring tiles. The length of the arrow represents the strength of the change, it points towards the strongest perceived change. Changes are normalized across all color maps to allow for a fair comparison. The pseudo-code in Algorithm 1 illustrates the computation.

We compute it by averaging the color distances between the center of the tile and all neighbors in ΔE as defined in Equation 4.1. We avoid false assumptions caused by extrapolation by com-

Algorithm 1 Compute difference arrows

```

Vector force  $\leftarrow$  [0, 0]
Color color  $\leftarrow$  tileColor(x, y)
for all Direction dir : directions(x, y) do
    Tile n  $\leftarrow$  tileModel.getNeighborFor(x, y, dir)
    Color ncolor  $\leftarrow$  tileColor(n.x, n.y)
    force += distance(color, ncolor) * dir
end for
return force

```

putting forces at border tiles only with a subset of tiles. As a consequence, arrows in border tiles always point along the border, never inside.

Interactive analysis of quantitative information More detailed information on the ΔE distances of an individual tile is shown when hovering it with the mouse cursor (see Figure 4.31 – right). The arrow glyphs for the tile at the cursor and adjacent tiles are removed and a detailed glyph is shown instead. As a result relative color distances of a tile to the closest neighbors can be analyzed in detail. The glyph consists of six arrows, each pointing to the center of one of the neighboring tiles (i.e. they all have equal length). The stroke thickness indicates the perceived color change. Detailed quantitative information for the point in the color map at the cursor position is listed in tabular form in a separate info panel. This pane is partly depicted in Figure 4.31 at the left.

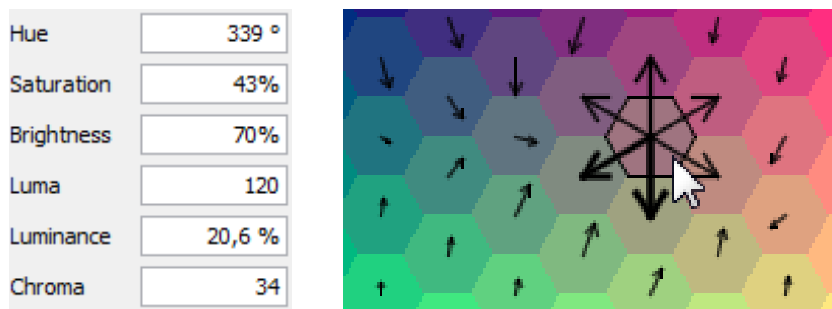


Figure 4.31: The detail arrow glyph shows individual difference to neighbor tiles. Quantitative information for that tile is given in the Info Panel at the left of the window.

The first two entries, X and Y, indicate the relative position of the mouse cursor on the color map in normalized coordinates. The next values represent the red, green, and blue component in the standard RGB color model that is used in many applications. The next three variables describe the color in the HSB color model, i.e. hue, saturation, and brightness. Hue is defined on a circular (connected) range from 0°-360°. Saturation and value are percentages.

CAM Lightness J is the brightness of a sample relative to the reference white. *CAM Hue* is the hue as defined in CieCAM02, which is not fundamentally different from other hue definitions,

but is well-aligned to human perception because hue linearity is one of its design goals. That is, a human observer is likely to perceive the same hue when given another sample with the same CAM hue but different brightness and/or chromatic content. *Hue quadrature* is a hue measure derived from hue where the values 0, 100, 200, and 300 correspond to the psychologically meaningful hues of red, yellow, green, and blue, respectively. *CAM Chroma* is the colorfulness of a stimulus as compared to the reference white, with 0 representing neutral colors. It is designed to be independent of lighting conditions. *CAM saturation* is the colorfulness of a stimulus as a proportion of its brightness. It is designed to be independent of the perceived brightness differences observable for different hues. The CIECAM02 Brightness (Q) and colorfulness (M) have not been included due to their strong dependency to the assumed viewing conditions. The viewing conditions are chosen based on the sRGB “typical” conditions and the guidance given in the CIECAM02 technical report (CIE 159:2004).

4.7.4.4. 3D View

In the 3D View, the visualization designer can assess how the shape of a single color map behaves in different color spaces. We take advantage of the fact that the RGB, CAM02-UCS (based on CIECAM02 as detailed above), HSB, and CIELAB color space can be spanned by three parameters. We provide 3D visualizations of the shape of a color map for every color space. These four visualizations are shown side by side in the 3D View. The shape of the color maps allows for an in-depth analysis of their properties. For example, a plane in the CieLab or CieCAM02 space indicates high perceptual linearity.

To transform the color map into the different 3D color spaces, we first sample the color map at regular grid coordinates. The color at the sampling points is then converted into the different color spaces. The Lab conversion is achieved by assuming sRGB primaries and an E reference white source as appropriate for self-luminous displays. The exact conversion routines can be found in the corresponding literature, which is comprised of CIE Publications (Lab: ISO 11664-4:2008(E) / CIE S 014-4/E:2007; CIECAM02: Technical Report 159:2004), the HSB proposal [Smi78], and the Luo et al. CAM02-UCS proposal [LCL06]. All of them have three components, and most of them can be used directly as spatial coordinates in a 3D surface plot. In order to represent the hue and saturation values of the HSB color space as spatial coordinates in 3D, we apply a transformation into polar coordinates where hue denotes the angle and saturation the radius. Consequently, the color space is a cylinder, not a cube as in the other cases.

One of the benefits of the 3D View is that visualization designers are supported in the identification of the color space that was used for the design of the color map. As an example, many color maps are constructed as planar cuts through the RGB cube. An illustrative example based on the HSB color space is the map of Alhoniemi and Simula as shown in Figure 4.32. It covers the entire hue and brightness ranges at a constant saturation. This is why it appears as a cylinder in the HSB visualization. The individual 3D visualizations allow for interactive manipulation of the virtual camera. The designer can rotate the plot and adjust the axis scaling.

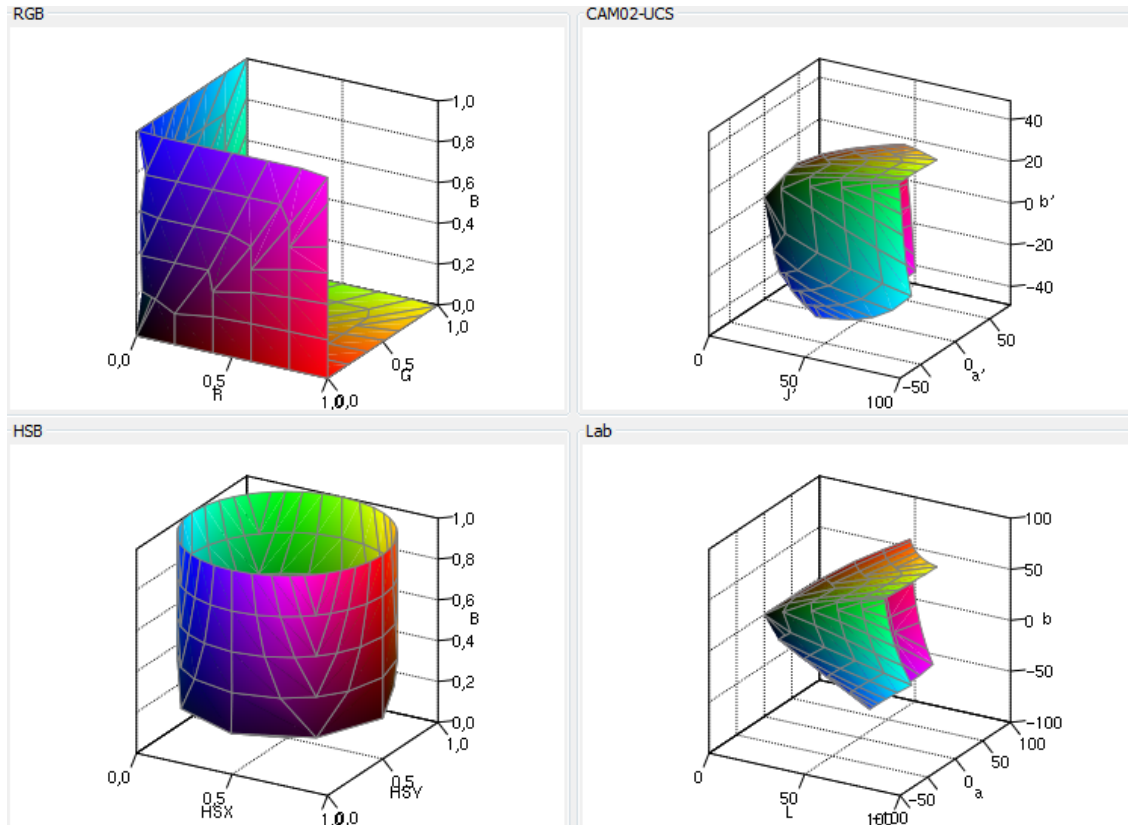


Figure 4.32: The 3D View of the map by Simula and Alhoniemi [SA99]: The color map is plotted in four different 3D spaces: the RGB and CIECAM02 cubes are at the top. The HSB space is actually a cylinder as the hue components defines a circle, the saturation its radius.

4.7.4.5. Color Sampling View

One of the most important quality aspects of a 2D color map is to faithfully represent spatial distances on the map with perceived color distances. This criterion is often called the perceptual linearity. Another important aspect is the number of distinguishable colors. The more colors a color map provides the more different information units can be encoded visually. We measure and illustrate the quality of both aspects in the Color Sampling View (see Figure 4.33).

Our approach first estimates the number of distinguishable colors based on the ΔE distance measure as described in Section 4.7.3. We solve an optimization problem trying to maximize the number of coordinates on the color map that fulfill the condition of a ΔE larger than a given threshold value t . The threshold is a user parameter and adjusts the minimum color distance in ΔE . A distance of $t = 1.0$ means that half of the observers are able to identify two colors as distinct. These points are depicted as white dots in Figure 4.33.

We compute the set of points using a circular sampling strategy: First, the center of the map is added to the result set. The algorithm then iterates on concentric circles around the center.

Each of these circles is sampled in regular intervals. The number of sampling points on the circle increases with the radius of the circle increases to guarantee an equal sampling density. Once the set of points is defined, pair-wise distances are computed. For each sample point, the color distance to all points in result set is computed in ΔE . A point is added to the result set if the distance is always smaller than t . We note that is algorithm produces merely an approximation, but a valid lower bound for the number of points. Assuming that the approximation quality is similar for different maps, it also allows to compare the number of colors.

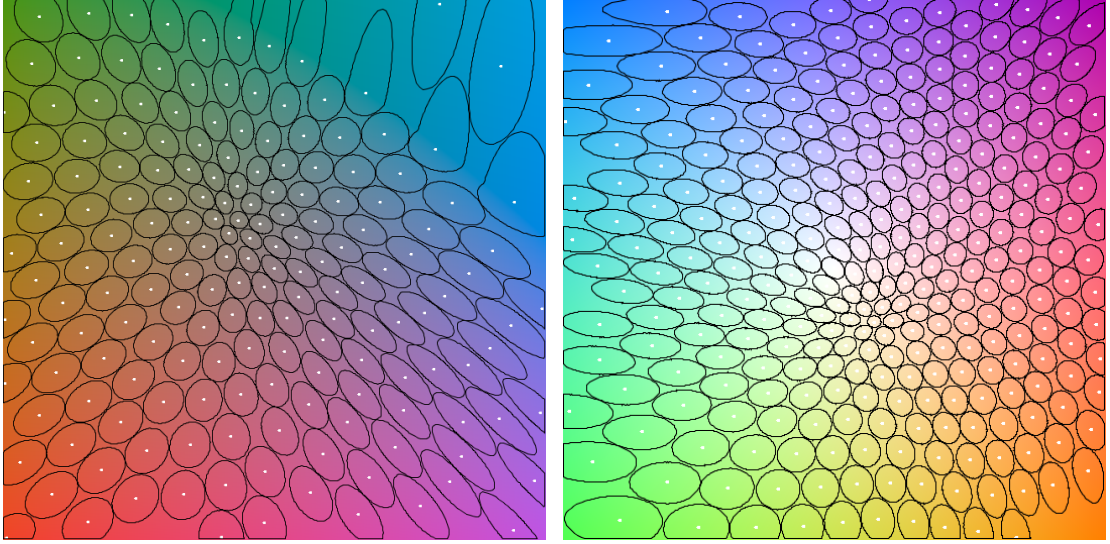


Figure 4.33: *The Color Sampling View for the color maps of Bremm (left) and TeulingFig3 (right): white dots indicate centroids of distinguishable colors, black surrounding polygons are isolines similar to MacAdam ellipses revealing information about local perceptual features.*

In a second step, our approach estimates the perceptual linearity of the color map based on the regions with similar colors. Based on the set of points that define distinguishable colors, the areas with a distance of max. $1/2 t$ to the central point are approximated. Starting at the center, the algorithm samples along a set of straight line segments at different angles. For every line segment, we estimate the point where the threshold $1/2 t$ is crossed. These crossing points are connected to an iso-line polygon. The resulting polygon is similar to the MacAdam ellipse [Mac42]. Major differences are that MacAdam ellipses are defined in the xy-plane and are real ellipses based on the minimum and maximum ratio between geometric distance and perceived color distance (as measured by a human test person).

The Color Sampling View depicts the coordinates of the result set as white dots and the surrounding iso-lines as black polygons. The number of white dots indicates the number of distinguishable colors. The shape of an iso-line allows for an in-depth analysis of local perceptual features. Circular areas indicate a high local perceptual linearity, because the change of color is identical for all directions. On the contrary, distorted shapes indicate a varying local perceptual linearity. An example can be seen at the upper right of Figure 4.33 – left. While most shapes

are rather circular, the upper right corner exhibits elliptical distortion. Individual divergences in shape can be identified easily by the user in this view.

The view also enables the visualization designer to compare different shapes. Variations in size indicate variances in the distribution of distinguishable colors. Thus, it can be seen that the perceptual linearity varies across the color map. In Figure 4.33, the color maps have 176 and 295 colors with a pair-wise distance of $5 \Delta E$. Interestingly, their distribution on the map is very different. In contrast, the map of Simula and Alhoniemi exhibits more than 600 colors.

4.7.4.6. Example Views

The usefulness of a color map for a visualization depends not only on intrinsic quality measures, but also on the usage context. Other visual environment parameters should be considered. With the example views, we support the visualization designer with test environments stressing color maps with other visual variables.

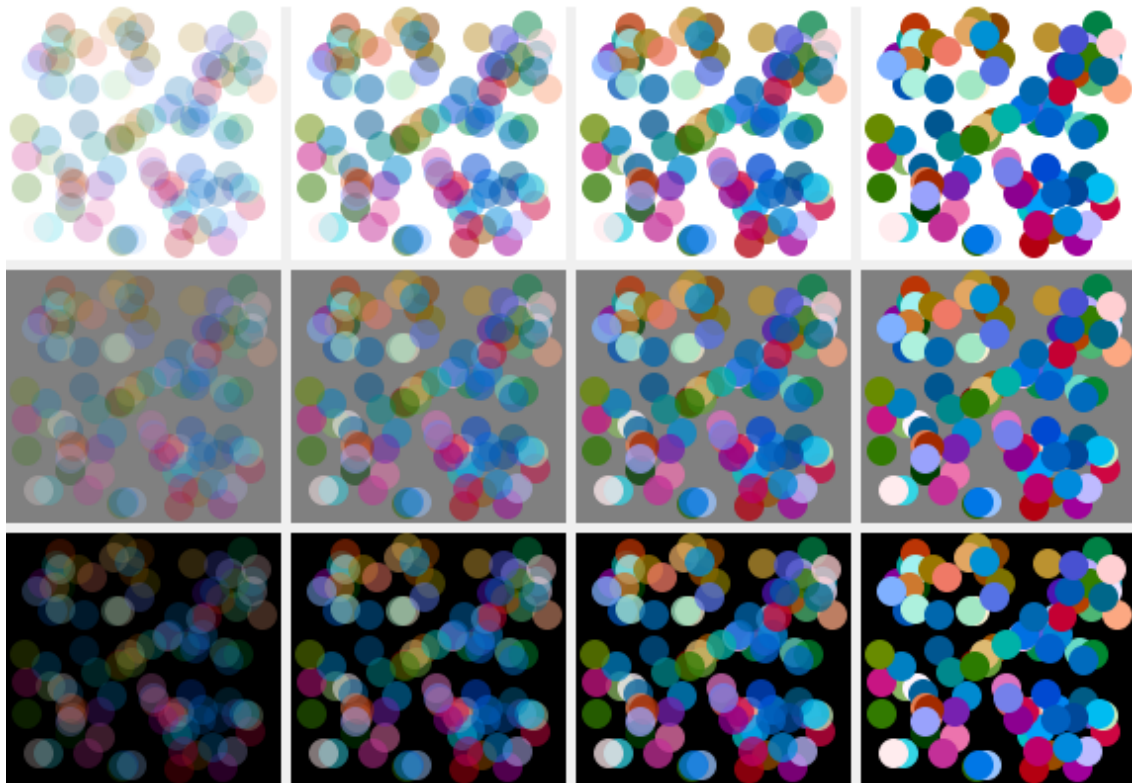


Figure 4.34: *The Points Example View. A set of 100 points is plotted on different backgrounds with different levels of transparency. The color of overlapping circles is blended.*

Point Set Example The first scenario illustrates the combination of the visual variables color (of the color map) and the position attribute. To that means, 100 equally-sized points with random

colors are aligned at random positions in a point-based scatterplot. The test environment is shown in Figure 4.34. The visualization designer is enabled to assess the applicability of a color map for spatial object distributions. For the sake of comparability of different tests the randomization is deterministic.

Additional visual aspects of possible interest are the transparency of the colored points and the interplay of the color map with the background color. To this end, we utilize the small-multiples pattern and duplicate the test setup by means of a 4×3 juxtaposition. The 12 test setups differ in the color transparency level (25%, 50%, 75%, and 100% alpha channel) and the chosen background colors (white, gray, and black). This grid is depicted in Figure 4.34.

As a result, the user can immediately see whether the chosen color map has a significant distance to the background and to which extent transparency can be used.

Text Overlay Example An important requirement of many visualizations is that text must be legible. Therefore, the readability of fine visual structures (such as printed text) on colored background is illustrated in the Text Overlay View, as shown in Figure 4.35. Visualization designers are enabled to assess the readability of the provided text snippets in a qualitative way.

We use black, medium gray, and white as contrasting text colors. The text is printed at different font sizes on a background that is generated from the color map. The two dynamic parameters (i.e. text color and font size) are varied in a small-multiples setup. The background of the test environments reflects the colors of the chosen color map.

Based on our experiments, the way the background is defined has a strong impact on the readability of the text. In particular, edges with sharp color contrast seem to distract the user's attention. To mitigate such effects, we use smooth (i.e. bilinear) interpolation of again pseudo-randomly selected color samples from the color map. This color is assigned to rectangles which are arranged in a two-dimensional grid in order to avoid irregular color changes. This view supports the user in comparing different environment variables in a text-based scenario.



Figure 4.35: *The Text Overlay View. Text is printed at different sizes in different colors on space-filling background that is generated from the selected color map.*

4.7.4.7. Use Case Examples

In this section, we demonstrate the usefulness of the ColorMap-Explorer tool. We show some of the findings that were made along an exemplary analysis workflow.

We conduct a scenario where a given color map is assumed to be ideal, be it on past experience or user preference. In this scenario, color should be used to encode information in a calendar-based visualization. We refer to Section 4.6 for a concrete use case of this colored calendar. Different variations of an example data set are depicted in Figure 4.36. The analytical task of the calendar view is mainly the comparison of individual (high-dimensional) data elements. Thus, the first important criterion is a large number of distinguishable colors to facilitate comparison tasks. The second criterion is perceptual linearity to adequately represent similarities of the data with color. Since the calendar grid is black, this color should be avoided.

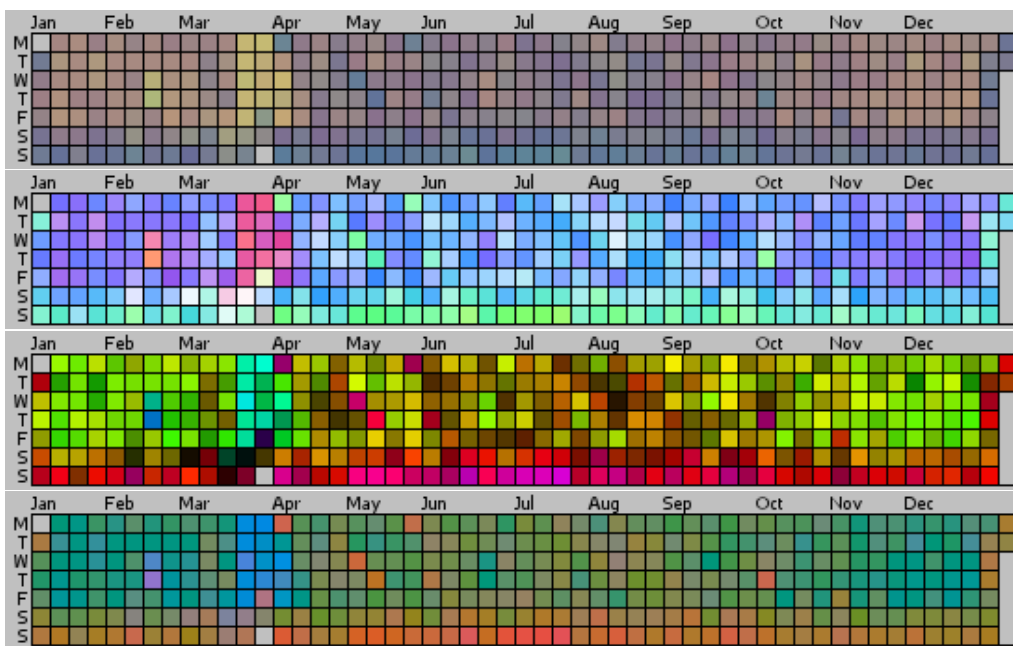


Figure 4.36: The calendar, rendered with different color maps. From top to bottom: BCP37, TeulingFig3, Simula and Alhoniemi, Bremm (regular).

The visualization designer uses the color map library that comes with the ColorMap-Explorer to experiment with different color maps. While some of the visualizations are more colorful than others, it is unclear to what extent similarity of items from the original data set is preserved. The ColorMap-Explorer can help finding out which color maps are suited for this task.

After starting the tool, the overview panel comes up and the visualization designer can view all 22 color maps at a glance as shown in Figure 4.27. Large discrepancies in the colorfulness of the different solutions become immediately visible. Since candidate maps should provide high color variations, the designer excludes maps such as BCP37 (the upper color map in Figure 4.36) or Robertson and O’Callaghan from further analysis due to their low colorfulness.

Based on the gained overview, the designer picks the most interesting color maps including the two variations of the color map of Bremm et al. [BvLBS11]. The Comparative View in Figure 4.29 shows them in juxtaposition. Although the number of colors is about average in the regular map, it is superior in the other scoring categories. Bremm et al. (regular) has a higher score than the stretched version in most categories. As a consequence, the stretched version is excluded from the candidate set.

In a next step, the visualization designer continues to the Decomposition Panel with remaining candidates for detailed inspection. In Figure 4.30 (Teuling Fig 2. [TSS11]), the second row shows that the red, green, and blue components increase across the map, but in different directions. The color map exploits all three RGB channels yielding a large number of distinguishable colors. However, the top-left view shows large arrow vectors along the rising diagonal. This reveals that the perceived color varies strongly leading to an inhomogeneous perceptual linearity in general. As a result, the designer rejects this color map for this task.

Aiming for high color exploitation, a colorful map such as Simula and Alhoniemi [SA99] is picked from the Overview Panel. The 3D view in Figure 4.32 confirms that the map covers large areas in the RGB and HSB color spaces. However, in the CieLab and CieCAM spaces the shape divergences strongly from a plane indicating a lack of perceptual linearity. As a consequence, the visualization designer concludes that the similarity of data items is not preserved well enough.

Our visualization expert returns to Bremm (regular) and opens the Color Sampling View. As can be seen in Figure 4.33 (left), the largest part of the map comprises small and well-shaped ellipses. However, in the top right corner an anomaly can be identified. The black outlines are unproportionally large and distorted. The color variation in this region is very low, leading to these large areas of similar color. Despite the fact that the map is mostly homogeneous, local features in the upper right corner hamper the homogeneity of the perceptual linearity. In contrast, in the map of Steiger et al. (see Figure 4.21) and in Teuling Fig. 3 [TSS11] (Figure 4.33 – right) this deficiency is less prominent. Repeating this comparison with other color maps reveals that (and why) Teuling Fig. 3 is among the best scored color maps with respect to the preservation of perceptual linearity.

In Figure 4.34, Guo's cone-shaped color map [GGMZ05] is plotted as a randomized Point Set Example. This color map has a very high color exploitation, which makes it easy to identify differently colored circles as such. However, some of the colors are very bright and hard to differentiate on white background, in particular at higher transparency levels. It is therefore better suited on dark backgrounds.

Looking at the Text Overlay example in Figure 4.35, the ColorMap-Explorer reveals that black text is fairly easy to read with the Teuling Fig. 3 color map [TSS11] – independent of the font size. On the other hand, gray and white are not ideal. One possible explanation is the similar brightness of foreground and background. Since bright text is not used in the target visualization, this is not an exclusion criterion.

The visualization designer concludes the decision support process of the use case. BCP37 was rejected due to the limited number of distinguishable colors. Simula and Alhoniemi has a high color exploitation, but the 3D View revealed a lack of perceptual linearity. With the Color Sampling View, Bremm (regular) was ruled out. The color maps of Teuling Fig. 3 and Steiger et

al. both exhibit high perceptual linearity and provide a fair color exploitation. The visualization designer therefore picks these two maps for the calendar visualization.

Having started with a set of preferred color maps, the ColorMap-Explorer enabled the visualization designer to reduce the number to two well-suited color maps. The decision support system enriched the decision making with qualitative and quantitative means.

4.7.5. Discussion

In this section, we showcased the ColorMap-Explorer, a tool for the visual exploration of 2D color maps. It gives an overview over many color maps that have been proposed in the literature on information visualization, provides different views for an in-detail analysis of strengths and weaknesses of color maps, and supports direct comparison, both visual and quantitative (i.e. based on explicit quality measures). Each color map can be exported as a high-resolution image. This enables the data scientist not only to find the best fit for a given task, but also to directly re-use the color map in other visualization software tools.

Current color map implementations are general purpose and independent from the data set. Similar to the optimization approach, color maps can be custom-tailored to specific data sets in order to achieve higher overall performance. The work of Mittelstädt et al. [MBS*14] already points in that direction. Integration such customized color maps in the explorer could help fostering that research area.

4.8. Exploring Simulation in Sensor Network Models

As soon as problems in the grid have been identified, the operators evaluate different options to compensate weaknesses through additional infrastructure or maintenance operations. But which options brings with most benefit in relation to expected cost and effort? Simulation is an important measure to estimate different properties of a planned network changes such as throughput and downtime. However, many parameters need to be adjusted to approximate real-world conditions properly. In this section we present a visualization system that visually supports and guides the analysis of (physical) network simulation problems (Contribution C₈). Automatic optimizers run as a black box giving an (locally) optimal result in terms of the underlying simulation model and parameter configuration. This is often not ideal for practical usage. Our system assists the user in the process of comparing different simulations to quickly achieve the optimal configuration in terms of user preference. It highlights differences between simulation runs and indicates which parameter modification leads to the best improvement. We expect that this results in large time savings for the domain expert while configuring the simulation system.

4.8.1. Visual Support for Simulations

Over the past years, urbanization has increased significantly, leading to continuous construction and extension of densely populated areas. Every time a new developing area is set up, not only houses and supermarkets are built, but also supply networks that provide gas, electricity and drinking water. These networks need to be laid out so that all customers are supplied at a minimum

of cost, time and effort. This is an optimization problem that needs to be solved. Several software-based solutions exist that simulate flow, cost and other quantities in such networks.

We have identified two tasks for urban planners and other, related domains: the first one is about defining a *new* plan for an optimal network layout based on a set of given constraints. The second one is about improving an existing network configuration to identify solutions that bring the largest benefit at the lowest cost.

Taking existing structures into account is more complex, but comes with the advantage that simulation results can be compared to physical conditions and real measurements. In either case, the simulation system must be configured to match real-world conditions as closely as possible. Only if this is given, the results of an optimization process are reliable and can therefore be trusted.

In general, the default configuration of the simulation system does not match the real circumstances of the user's task domain. She therefore needs to iteratively tweak the parameters until a desired target result is achieved. In this process, it is important to compare different simulation runs to find out which effect a parameter change has on which target variables.

Based on a formal description of the simulation parameters, optimizers can find a configuration that is "best" for one or more variables in a mathematical sense. However, the mathematical optimal result is not always the optimal result from the user's perspective. The user might have a target state in mind, but it is not always clear how to converge from the current state towards the target state: The optimizer may provide an optimal solution, but it will usually not be able to determine whether the *transition* from the current state to the optimal state is feasible. This feasibility can refer to *explicit* constraints that are imposed by the simulation model, for example, the conservation of momentum and energy, or box constraints that are taken into account during the optimization. But it may also refer to *implicit* constraints that are not part of the simulation model. For example, whether a modification or restructuring of a supply network is feasible within a certain time constraint or not needs to be clarified.

Another problem with this is that these simulators run as a black-box and the user is confronted with a large number of input- and result variables, often in tabular form. An appropriate visualization of the network with interactive analysis properties is key to an in-depth understanding of the optimization problem. In the scientific community, this is sometimes referred to as *Computational Steering*: an iterative, interactive process that intertwines user and machine to achieve better results in less time [WBD00]. Closely related is the field of *Visual Analytics* where iterative, alternating human and automatic analysis are combined.

4.8.1.1. Overview

We present an analysis system to overcome the aforementioned limitations: It consists of several, closely coupled views with interaction support, which can help the user to gain a more intuitive understanding about the inter-dependencies between the input- and output parameters and the behavior of the system as a whole:

- A network visualization that shows the topological and geographic layout
- An simulation browser that provides the configuration history tree

- Drill-down support that adjust the level of detail depending on the zoom factor
- Visual-interactive configuration of simulation parameters

As a result the user can interactively explore different network simulation parameter settings. The *global* effect of changing individual design parameters is visualized in the parameter configuration view. The *local* effects for the individual network nodes are shown in the network view, and changes can be compared on a per-node basis.

The user is supported in the exploration by a visual history of parameter changes. The manual analysis process is documented in the history view, supporting an active, user-steered exploration of the design space.

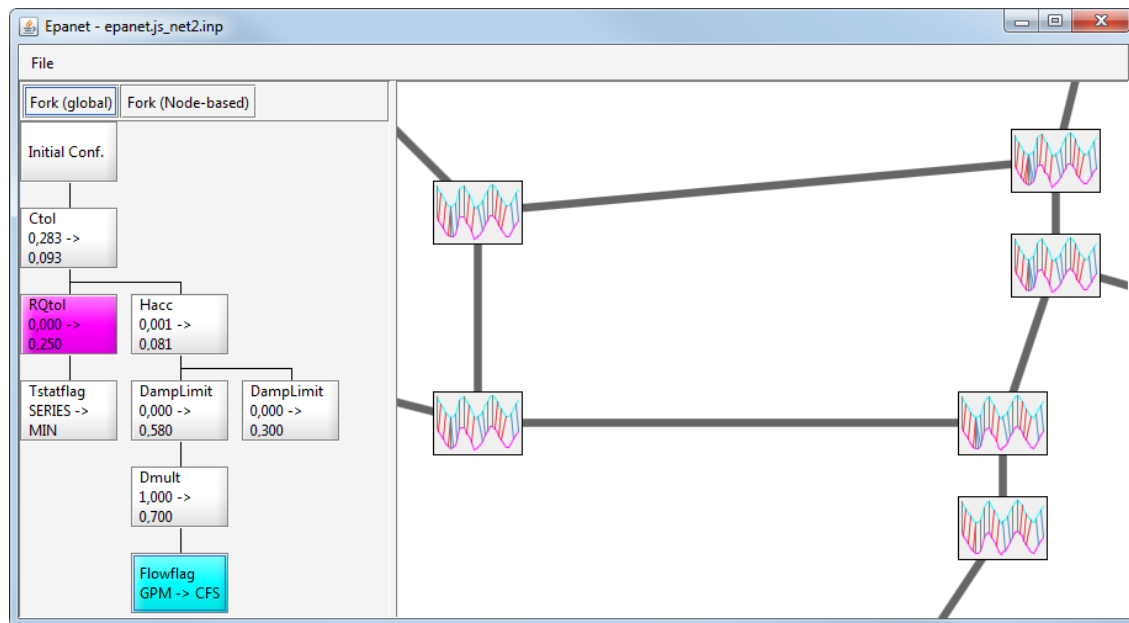


Figure 4.37: The main window of the visualization system with its two views. The Simulation History View on the left illustrates the different branches of the simulation parameter changes. The Network View on the right gives an overview on the simulation results distributed in the network topology.

4.8.1.2. Simulation History

In the work of Afzal et al. the impact of user decisions is shown in a simulation of epidemic spreading [AME11]. Similar to our history view, one of the views illustrates the decision history the user made during the simulation and its impacts on chosen target functions. Unger et al. describe a system to visualize different statistical experiments of biochemical reaction networks [US09]. The focus of this work was on comparing different simulation runs. Compared to our approach, it focuses on the results in terms of minimal and maximal extent, not on differences

in time or in the parameter space. In the work of Brodlie et al. [BPW*93] a history tree is used, again similar to our approach, to enable the user to backtrack parameter value changes.

A recent survey of methods to “visualize alternatives in multiple criteria decision making problems” can be found in the publication by Miettinen et al. [Mie14].

4.8.1.3. Parameter Space Exploration

The exploration of parameter spaces for complex simulation models is a task with many practical applications, and a large variety of tools supporting this exploration and analysis process exist.

A visualization for refining a coarsely sampled subspace of the parameter space was realized by Luboschik et al. [LRHS14]. Heterogeneous information between adjacent scales of the parameter and time space of the simulation results is used to enable the user to refine the simulation results at a reasonable level. Their work, however, does not focus on networks. Matkovic et al. [MGJH11] present a visual-interactive system for simulation analysis. They define a fixed set of *control parameters* and *response parameters* for the simulation. Similar to our approach, the control parameters are sampled with a certain number of variations, but the simulation runs are pre-computed.

The *HyperMoVal* tool, developed by Piringer et al. [PBK10], also aims at the optimization of engine construction, but more focused on the validation of regression models. The work identifies several key tasks, like comparison of results and models, as well as quality estimation and assessment of plausibility. We picked up these ideas by differentiating local and global parameters and their effects in the simulated network.

The concept of *Design Steering*, as a special case for the *Computational Steering* during the design process, was investigated by Wright et al. [WBD00]. Their system allows the user to manually or automatically navigate in a six-dimensional space of design parameters, while building a trajectory that also served as a history of previous design attempts. In our work, all parameters are projected into 2D space for better comparison and also to allow for a higher number of parameters.

Another application of parameter space exploration was examined by Bruckner et al. [BM10]. In their case, the parameters are shown as circular parallel-coordinate plots, because the main focus of this work was not on the visualization of the *parameter space*, but on the visualization of the *simulation space* and its time characteristics.

An approach for parameter space exploration that is agnostic from the application domain was presented by Berger et al. [BPFG11]. It covers visual guidance to quickly identify interesting parameter regions. This is mainly achieved by sampling parameter values in a certain range and with a certain step size. As a result, the sensitivity and general effects of changing individual parameters can be estimated. This approach is similar to the *Parameter Configuration View* that we describe in Section 4.8.2.5.

4.8.2. Concept

In this section we will first present the data and the simulation system we are working with. We will then outline the idea behind of our visualization systems and discuss the main views in more detail.

4.8.2.1. Data

We start with the simulation data for the visualization system. The underlying data structure is a (geographical) network with one or more time-dependent variables for each graph node. This could be, for example, water pressure sensors in a network of pump stations or the time-varying demand of water supply at a particular node.

Using a simulation system, we can compute different quantities for the nodes in this network. It can be modeled as a function that takes a set of global parameters and parameters for each node. This input data is transformed into a time-based series of output variables. The data in this case stems from purely artificial data sources, namely from the simulation. Therefore we do not have to take into account measurement imprecision or missing values. It is, however, important to either specify valid parameter ranges or identify illegal configurations. Some parameters might have explicit minimum and maximum values, while some have cross-dependencies to other parameters. In the latter case, this specification is difficult and it is often easier to deal with invalid results. We will describe in Section 4.8.2.5 how invalid simulation results may be treated in the visualization.

4.8.2.2. Visualization System Overview

The visualization system consists of complementary views and a supporting configuration dialog. The main window consists of the Network View and the Simulation History View (see Figure 4.37). This window is complemented by the Parameter Configuration Dialog (see Figure 4.40). All views are discussed in detail in the order they appear in a typical workflow pattern in the following sections.

4.8.2.3. Network View

The *Network View* that gives an overview on the simulation network, as shown in Figure 4.38. In our use cases, the number of nodes is rather limited and usually associated with a geographical position. The natural choice for the visual representation of these networks is a *node-link diagrams* with a geo-referenced layout. This makes it easier for the user to identify the nodes and map them to the real entities (pumps, tanks, etc). The user can thus quickly get the 'big picture' of the layout, including domain-specific elements such as tanks, pumps, etc. Nodes are represented by filled rectangles, the links in between are indicated by straight line segments.

Although the nodes are typically referenced geographically, we do not show a map in the background to keep the visual attention on the network. Geographical layout is not the focus in this application – we therefore chose a plain white background to avoid any kind of distraction.

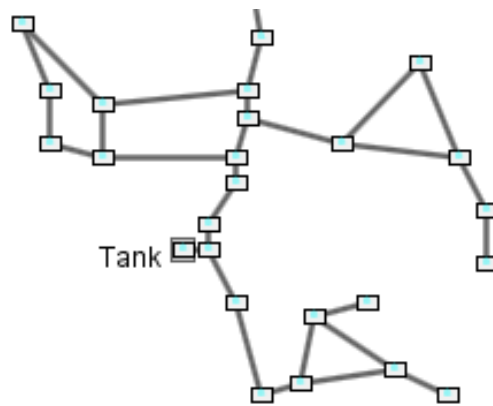


Figure 4.38: *An initial overview over the simulation network.*

The user can interact with this view with the mouse, similar to well-known online web map services. Using a virtual camera, the user can pan the view by dragging the mouse cursor and zoom in and out using the mouse wheel.

4.8.2.4. Forking

In case that the user is not satisfied with this initial simulation parametrization, a new simulation run can be defined. This new parameter configuration is based on the current (in the first iteration the initial) setup, but with slightly different settings. Akin to the natural phenomena and the operation in software revision control systems, we call this operation “fork”, because it describes a deviating branch.

There are two different modes: a fork can modify global simulation parameters or based on a single element (i.e. node or edge) in the network. Forking the global configuration is always permitted while the other mode requires the user to first select the element that should be reconfigured. This is done by simply clicking individual nodes or edges.

4.8.2.5. Parameter Configuration View

The forking operation is performed using the *Parameter Configuration View*. The parameter set of the underlying simulation is displayed as a series of GUI components based on the data type of the parameters: Categorical values are mapped to combo boxes and Boolean flags to checkboxes. Numerical values are represented as spinning text fields. The mapped parameter model is illustrated in Figure 4.39.

The user can then modify the parameter configuration using the UI controls. Our tool assists the user with an interactive preview: The effect of changing parameters with continuous ranges is shown in aggregated form (see Figure 4.40). Starting from the current configuration, changes in individual parameter values are simulated in a background process and displayed as soon as the value is computed.





ChemName	Fluoride
ChemUnits	mg/L
 Climit	0
 Ctol	0,283
 DampLimit	0
 Dcost	0

Figure 4.39: The parameter configuration model is converted into a set of UI component to enable the user to interactively modify values within valid bounds. Parameters that are continuous are simulated and therefore annotated with a small legend icon.

The process iterates on every parameter individually, starting from the current value towards both the minimum and the maximum value. The parameter range is sampled at equidistant intervals, but more sophisticated strategies are also feasible. As the individual simulations are not dependent on each other, they can be run in parallel, which is key to the interactive display of the view. The view is continuously updated as new values are computed in the background. The tasks for the computation of the simulation results are scheduled in an order that corresponds to a breadth-first search in the parameter space: The tasks with small changes in each parameter are scheduled first, and the tasks for the simulation with the largest parameter changes are scheduled last. This has the effect of the result paths “growing” starting at the initial state. This allows the user to quickly evaluate the most promising parameter changes without having to wait for all simulation runs to complete.

We can define an arbitrary projection of the result space into the screen space. The simplest is to pick two variables from the result space and use them as x and y coordinates to visualize the outcome of the simulation in a scatterplot-like display. Axes and a grid are drawn in the background to provide information about the actual parameter values and the currently displayed area of the projected result space.

Individual parameters are assigned to unique colors, shown as a legend next to the corresponding UI controls. The most important decision criterion for the set of used colors was to be discriminative on white background. This is why we chose the set of qualitative colors as defined by Harrower and Brewer [HB03b].

Depending on the zoom level, individual results are depicted as dots or small circles. Results from consecutive adjustments on the same parameter value are connected with a straight line segment in the color of the corresponding parameter to indicate their relatedness. We refer to these connected sets as *result paths*. We prefer straight lines over splines, because they clearly indicate that no information is available in between two consecutive points.

Hovering the mouse over any of the results provides a tooltip with information about the name and value of the parameter that was changed to achieve the result, as well as optional information about the actual simulation output. Additionally, the corresponding path is painted with a thicker stroke, while the others are painted thinner. This allows the user to focus on the analysis of the behavior of a single parameter even when multiple parameter paths are displayed or have a similar shape.

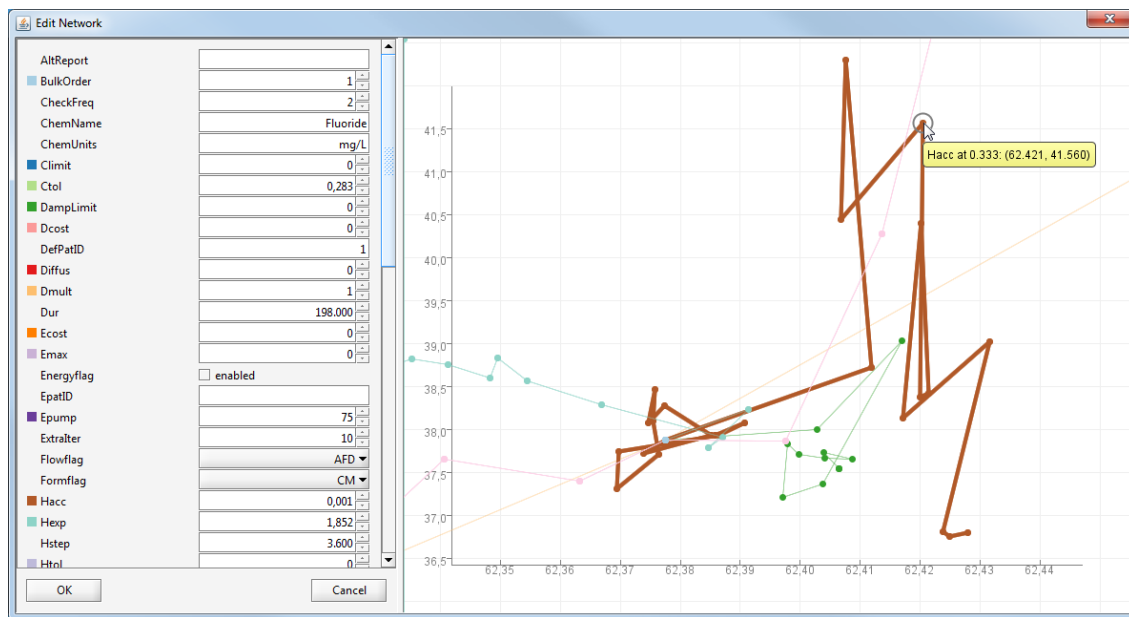


Figure 4.40: The parameter view gives a first impression of the effect a single parameter change has. Each parameter is represented with a unique color, points along a line represent different parameter values. The point location is derived from two global target functions.

Depending on the underlying simulation system, information about the feasibility of a given parameter configuration can be incorporated in the output. One option would be to simply omit the corresponding result paths. In other cases, it could be preferable to paint the invalid results in a pale color, so that they are not distracting the user from the feasible results. This would still allow her to see the connections between the results for the case that one parameter was only passing through an infeasible region during the interpolation.

Depending on the validity of the simulation results, the stability of the simulation and the chosen projection of the results into 2D, the points representing the results may be at extreme positions. For this reason, we chose to not perform an automatic normalization, zooming or clipping. Instead, a virtual camera – similar to that one in the main view – was introduced to support interactive panning and zooming. This camera additionally allows to independently zoom the horizontal and vertical axis. The codomains of the result space are both abstract and independent from each other, so the user can freely choose the region of the result space that is to be displayed, focusing on the result path that he is currently interested in.

To summarize, the Parameter View enables the user to view the effect of parameter changes in a single and quick overview. She can find out which parameter changes brings the largest improvement based on the current configuration or tweak a mathematically optimal solution to other, user-defined preferences.

4.8.2.6. Simulation History View

Once the user has decided on which parameter to change, the corresponding values are set and the user returns to the main window. This completes the forking operation and the *Simulation History View* is updated accordingly. In this view, an overview over previously computed simulation runs is given in a tree-based hierarchical layout. It starts at the top-left corner with the initial configuration; directly derived configurations are placed one level (on the vertical axis) below. Configurations that stem from the same parent configuration reside on the same level. The first child element is always put directly below the parent element; its siblings are added to the right. Related configurations are linked by elbow connectors. See Figure 4.41 for an example.

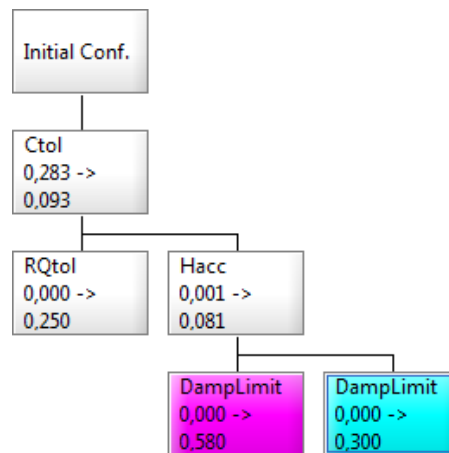


Figure 4.41: The History View depicts all simulation parameter configurations that were derived from an initial starting point. This tree-view shows the history in terms of configuration changes. It can be split into different branches and any two configurations can be shown and compared by selecting them (cyan and magenta).

Two of the configurations can be selected by the user to compare them in detail in the Network View. We chose the colors *cyan* and *magenta*, because this pair fulfills the following criteria: They have similar brightness and saturation (almost 100%), they are discriminative even for color-deficient users and they do not have an intrinsic attribution.

4.8.2.7. Network Node Glyph

When the user has selected two configurations, the Network View allows comparing them on a per-node level. The content of the node glyphs depends on the zoom level and the user selection. Figure 4.42 depicts a node with two time-varying output variables as an embedded line chart. The two line charts have the same color as the selected configurations in the History View. This enables the user to map the line charts to the corresponding simulations. Initially, a simulation based on the default settings is run on startup, and the results are then shown in the node glyphs. When two different simulation runs are selected for comparison, they are drawn in the upper and

lower half of the glyph, respectively. Zooming in increases the granularity of the displayed time series.

The third part of the glyph is the display of temporal shifts between the two simulations. We use the approximate Dynamic Time Warping (DTW) approach by Salvador and Chan [SP04] to compute the distance between two time-series data sets. We are, however, not only interested in the total distance, but also in the actual computation of that distance. In contrast to the Euclidean distance, DTW aligns sequences of the data based on a cost function to respect shifts along the time axis. These shifts are displayed in the node glyph by connecting those nodes with the smallest computed distance. If the corresponding indices are identical for both configurations, a thin vertical black line is drawn. Different indices indicate distortions and the line is drawn thicker to raise more awareness. The line color is either *red* or *blue*, depending on the direction of the shift.

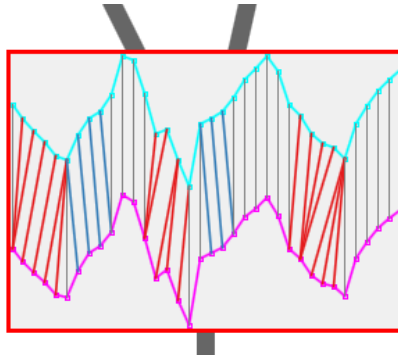


Figure 4.42: Comparison of simulation results for a single node. One of the output variables can be displayed in the node glyph (here: pressure over time) The color of the line chart corresponds to the color of the selected configurations. Shifts in the data are highlighted by connecting lines. The line color indicates the direction of the shift.

In this section, we described the different views and their use in detail. The “Fork” operation creates a new simulation based on the current one. This new branch can be configured using the Parameter View. The Simulation History keeps track of changes while Network View enables the user to analyze differences in the network in detail.

4.8.3. Case Study

We performed a case study on the basis of a water distribution piping system in order to evaluate the applicability of our system for optimization tasks, and in order to validate the workflow.

4.8.3.1. Setup and Task

The network structure for our test case is taken from an anonymized small real-world data set containing 25 nodes, one tank, one pump and 41 connecting pipes. The public domain simulation software EPANET [US 08] was used to simulate the node pressures, the flow in the pipes and the height of the water in each tank. It was originally developed by the US Environmental Protection

Agency (EPA), later continued and ported to Java by the company Addition [Add12]. This software was used to simulate the network for 24 hours in 10 minute sample intervals. Computing such a simulation took less than one second on a 2014 commodity computer.

The goal of the user is to modify the network in order to minimize the fluctuations in the node pressure and increase the stability in the network. He wants to achieve this by replacing a tank in the network with a different one.

4.8.3.2. Workflow

The first step is to load the simulation input file containing the network information and the default simulation settings. The *Network View* gives the analyst an overview on the infrastructure. Zooming and panning and the node labels allow quickly locating and focusing on the tank. Selecting the tank allows for forking the initial simulation configuration based on different parameters for the tank.

Therefore, the user opens the *Parameter Configuration View*. Variations of the simulation settings are computed in the background, and displayed in the view. Based on the shape of the result paths, the user decides to increase the size of the tank by 50% to have additional reserve capacity. After changing this parameter, he returns to the main window, where the new configuration is displayed as a new child node of the initial configuration in the *Simulation History View*.

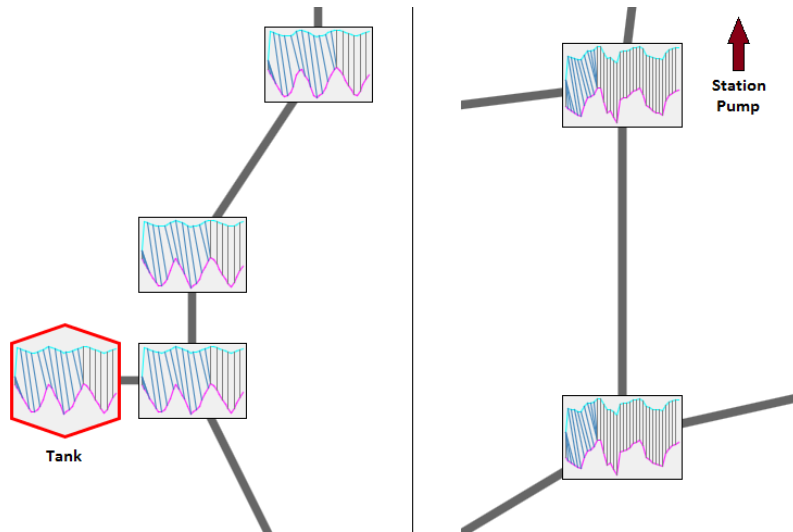


Figure 4.43: The nodes that are close to the modified tank are strongly affected by the change while those that are further away and close to the main pump (right image) are almost identical to the previous simulation.

Zooming into the Network View increases the level of detail of the information displayed in the node glyphs to show the time-dependent variables. Selecting two nodes in the Simulation History View reveals the temporal shifts between these properties in form of a dynamic time warping comparison. This shows that the node pressure changed, but there still are large fluctuations in

the node pressure. So he continues his exploration with another parameter. The Parameter Configuration View showed a strong influence of the tank base height on the node pressure, indicated by the result path covering a wide range of possible values of the objective function (see Figure 4.40). So instead of changing the tank size, the user now decided to change the base height of the tank. Selecting the original and the new node in the Simulation History View again visualizes the effect of this change on the network: The fluctuations in the node pressure of the nodes close to the tank become smaller. With increasing distance to the tank, the effect becomes weaker (see Figure 4.43).

The actual goal that has been achieved with this modification – namely, reducing the fluctuation of the pressure – could also be achieved with a completely automated optimization. However, there are several advantages when using the visual-interactive approach: First of all, it is not necessary to find a mathematical description of the design goal that can serve as an objective function for the optimizer. The user can gain a more intuitive understanding of the simulation model. Additionally, the process of how the optimization goal was achieved is documented in the history view.

4.8.4. Discussion

In this section we demonstrated a visualization system to visually explore different parameter configurations for time-dependent network simulations. Using the combination of Network View, History View and the Simulation Configuration Dialog, the user is enabled to configure the simulation system and keep track of these changes. The influence of individual parameters on the target function and their effects on the time-dependent output variables can be evaluated visually. This may lead to a deeper understanding of the network structure and interdependencies between properties of the individual network nodes, and a more focused and goal-oriented exploration of the parameter space while analyzing and optimizing the network model.

In our current implementation, the edges are represented as simple lines, suggesting an undirected graph. In many real-world applications and tasks, like the water supply networks that we are examining in our use case, the networks are actually *directed graphs*. An appropriate representation of the edge direction might serve as an additional visual aid in order to understand the structure of the network as well as the effects that modifications in the network will have. Holten et al. discuss representations for directed edges in terms of precision and readability [HivWF11]. The focus of this part is on the support of the analysis workflow itself.

The most important limitation for interactive exploration of simulation data is the responsiveness of the simulation. In those cases where simulation takes a second or longer, the use of simplified model that is faster to compute could be considered. Similarly, sophisticated sampling strategies could help to reduce the number of runs or increase the quality of the result path.

The parameter visualization view shows two global output variables. When more than two dimensions are required, different visualizations such as a scatterplot matrix might be more appropriate. On the other hand, the parameter changes that affect only parts of the network could be of interest. Then, a small parameter visualization could be shown inside the node glyphs.

4.9. Summary

This chapter discussed the *Visual Analysis of Sensor Networks*, a particular group of networks that combine time-series data with graph data. With the increase of installed sensors in many aspects of both daily life but also industrial production and monitoring, the analysis of this type of data is highly relevant for many practical problems.

Of particular interest are electric grids, not only since some European countries decided to enforce the use of renewable energy. The shift of interests towards a bigger amount of smaller plants distributed over large areas drives the innovation in this domain. This trend leads to a significant amount of innovation, but also causes drastic changes in the grid. Some of approaches that are discussed in this chapter use this scenario as an application example, but their applicability holds also for other types of sensor network data.

In analogy to Chapter 3, a set of explicit requirements is defined in Section 4.2. These requirements aim to cover the ability of a data analyst to successfully work with an analysis tool for sensor networks. The overarching goals here are to gain an overview of a potentially large network, identify anomalies without explicit declaration in space and time and to provide support for planning operations. Section 4.3 covers the fundamentals of sensor networks; later sections refer to this part for the basic definitions. Previous approaches and related scientific work is discussed in Section 4.3. This section is split into two parts. The first one covers automated approaches, which are typically backed by expert systems or similar algorithms. The other part deals with approaches that involve the human user through visualization and interaction metaphors. Highly correlated is Section 4.4, which explicitly discusses a variety of approaches for the visual layout of sensor networks.

Based on the literature review, several missing points have been identified. Sections 4.5 – 4.8 present different variations of a visualization system that explicitly deal with these shortcomings. The approaches that are discussed in these sections aim to fulfill the aforementioned requirements. We will now check in how far this has been accomplished.

- Requirement 6 (Relative Directions): This requirement can be fulfilled by a collection of approaches. Section 4.4 gives an overview over the state of the art for direction-preserving layout strategies.
- Requirement 7 (Data Sources): The approach that is presented in Section 4.5 integrates heterogeneous data sources such as weather data or ICT coverage in a multi-view and multi-perspective analysis environment. To give a real-world example, analyst can thus correlate electricity production with solar radiation to assess the effect of solar panels on the electric grid.,
- Requirement 8 (Overview): An overview visualization was provided in all of the presented approaches based on the metaphor of a virtual camera. Mouse-based panning and zooming adjusts the visible viewport as needed. Zooming in and out also adjusts the level of details that is employed to depict nodes in the network. We refer to Sections 4.5, 4.6, and 4.8 for different variations.

- Requirement 9 (Automated/Manual Analysis): In Section 4.5, a combination of expert system and multi-view environment is presented. This enables the operator to gain insight in the modus operandi of the algorithmic analysis pipeline and adjust it as needed.
- Requirement 10 (Compare Sensors): The calendar view that was described in Section 4.6 enables the analyst to compare different sensors. Even more so, the node glyph of the approach that was presented in Section 4.8 provides a side-by-side comparison inside a single glyph.
- Requirement 11 (Identify Anomalies): This requirement is fulfilled by the approach that was described in Section 4.6.
- Requirement 12 (Planning Support): As described in Section 4.8, a “What-If”-Analysis can be performed to assess the sensitivity of different parameters in a network simulation. The effect of modifications to the simulation parametrization is shown for every node.

As can be seen from the enumeration, all requirements have been met. In this chapter of the thesis, a series of approaches was presented and discussed that use visualization and interaction to support the analysis of sensor networks. From a practical perspective, this is useful for the management of structures such as electrical grids or water supply networks. Operators and managers are provided with tools to explore networks, find anomalies in time and space and evaluate network simulations for planning and maintenance operations.

5. Conclusions and Outlook

The machine-based recording of observations and environmental measurements exist for decades. However, with the exponential increase of computing power and resolution of sensing devices, the amount of data has become overwhelming. Computers need explicit processing instructions in order to extract valuable information from it. As a consequence, the analyst needs to know beforehand what the machine should do to get the expected results. Whenever this is not possible, a tighter integration of human understanding in the computing process is necessary.

The focus of this work is on network data, and more specifically on sensor networks. This special kind of network integrates time-series data and graph structure on a per-node basis. The overarching goal of this work is the support of the user in exploratory search and related analysis tasks.

The thesis started with a general introduction and motivation in Chapter 1. It briefly outlined the advantages of combining strengths of man and machine and gave an overview of the thesis' contribution to the scientific domain. Chapter 2 presented the foundation on which visual analysis tools are built upon. It covered a short history of *Information Visualization*, sketched the most relevant models and tasks before going into more detail on data types and visualizations. This also included interaction techniques and their integration in the *Visual Analytics* process. In Chapter 3, several approaches were described in the frame of explorative network analysis of general graph structures. These approaches follow a common strategy, namely to focus on local areas that are of specific interest to the data analyst. The remaining part of the graph is not shown, but hinted by visual signposts. Successful exploration of such networks requires the preservation of the mental map of the analyst. This, in turn, requires that changes in the layout can be tracked by the user and orientation guidelines that point towards interesting or relevant parts of the graph outside the visible region. A stable and deterministic graph layout plays a key role for this. All of these aspects were discussed in detail in this chapter.

Chapter 4 picked up these ideas on general graph layout and navigation and applied them to the domain of sensor networks. It covered an introduction to the different data types, followed by a review of layout strategies that preserve relative directions or even exact position coordinates. Such properties are often useful for practitioners as many physical sensor networks have a planar layout in geographical coordinates. This helps orienting in the visualization as it reflects the physical layout. On that background, different visualization and interaction approaches were presented. They discuss the integration of automatic assessment of networks events (e.g. through expert systems) and heterogeneous data sources (e.g. weather, ICT coverage). As a consequence, the analyst is enabled to inspect the network with respect to anomaly detection in space and time. As soon as problems have been identified, the analyst might want to plan countermeasures. Tools that work with "What-if"-simulation provide support to anticipate the effect of different options on the network.

In summary, the series of approaches that were discussed in the frame of this thesis support the analysis and management of sensor networks. With the set of contributions that was presented in the first part the analyst is enabled to browse through graph data in efficient, yet effective way. The visual representation of sub-graphs that have already been explored before are re-created in a deterministic manner, enabling the analyst to recognize known areas. Several other closely linked techniques aim to preserve the user's mental map by creating smooth layout transitions between consecutive steps of the exploration process. With the help of a signposts metaphor, interesting other parts of the graph are indicated but also support the user's orientation in analogy to real signposts.

In the second part of this work specifically deals with a combination of graph data with time-series data, where each of the nodes is linked to one time-series. We refer to this data as sensor networks. A multi-faceted visualization system for the analysis supports the effective monitoring management of these sensor networks was presented. With its help, the analyst can quickly gain an overview on the network itself, but also see linked data sources, for example to assess the influence of weather. Similar to existing systems, event data can be processed using a rule-based automatic approach. However, our system goes beyond the state of the art by integrating this automated process into the visualization system, allowing the analyst to open this "black box" and monitor the process. One major short-coming of automated systems is that they can deal with explicit facts only. We presented an approach for the analysis of temporal patterns in sensor network data to enable the analyst to identify nodes that behave differently from the others without the need to explicitly state what "similar" or "different" means. This even works in the temporal domain and in different granularities, enabling the identification of daily, weekly or seasonal effects. One application scenario is the monitoring of the electric grid where domain experts can use this knowledge to find weaknesses in the network, identify the best point in time for maintenance work on cables and see the effect of solar plants on the flow in the network. As soon as a weak node or link has been identified, our system supports the experts in finding the best strategy to compensate by integrating simulation in the visualization. Different scenarios can be run and compared on a per-node basis, thus giving very detailed feedback on the expected effect of the planned changes. The use cases of this work were located in the domain of supply networks, in particular electricity and fresh-water. Since no domain-specific knowledge is required, the system is equally applicable to other domains such as logistics, transportation and industrial manufacturing. As a whole, the system supports the analyst in different stages of network operation: daily operation, long-term analysis, maintenance and grid expansion.

5.1. Future Work

The initial motivation for our work – the increase of data beyond the ability to process it manually – will become even stronger in the future. We group possible directions for future work in two categories. In analogy to the overall structure of this thesis, the first category specifically deals with Local Graph Views, in particular with the results of the user study. The second category discusses future work for the sensor networks.

5.1.1. Local Graph Views

In Chapter 3, a list of requirements for the successful exploration of graphs on the basis of local graph views that was enumerated. While we realize that all of them have been met, we also see possible further improvements of the presented approaches. We plan to pick of some of the ideas from the initial user study on the signposts approach to further improve visualization and interaction means. Future work includes better support for the history of the exploration.

The dynamic layout approaches improve the stability during the exploration process. Using stitching to combine pre-computed layout patches even ensures that the exploration is deterministic and structures can be recognized. However, the strength of layout stitching is also a weakness: interpreting a graph as a point cloud without structure allows to create a unified layout of both but at the cost of its structure. We are convinced that the interpretation of layouts as images features a plethora of concepts and approaches just waiting to be transferred and applied to graph layouts.

In particular for large graphs, navigating through the entire data set is a cumbersome feature and makes it difficult to identify larger structure. Using aggregation techniques like clustering, the graph complexity can be significantly reduced. However, the overall appearance of the network can be quite different. The ability to preserve shape across different levels-of-details appears to be both an important, but also an achievable goal. Therefore, the next logical step for us is to apply this approach to hierarchical navigation concepts.

5.1.2. Sensor Networks

In a similar manner, different requirements for the analysis of sensor networks have been identified in Chapter 4. Again, all of the mentioned item have been addressed with one or more of the presented contributions. Nonetheless, we also see room for further improvements of the presented approaches.

At present, the visualization environment that wraps around the expert system is predominantly designed for monitoring purposes. It could be useful to extend its functionality to a *modeling tool*. This would require that the domain expert interactively modifies the model and the rule collection. The operator could update the model whenever changes in the physical world are made.

That aside, a major aspect of the visualization is that the data is univariate, i.e. only a single series temporal data is attached to a node. Future work includes the extension to multi-variate data sets. A challenge is to integrate both multi-variate data and the time domain in one similarity model. Moreover, an appropriate visual representation of the data is required, in particular for the glyph that represents clusters of similar elements.

Also, the inclusion of semi-interactive clustering to better adjust the representation patterns is of interest. This could be achieved by iterative split and merge operations or through adjustment of the distance functions and the clustering parameters. Another aspect is to extend the system to also support live monitoring. As of now, the major restriction is the insertion of new measures in already existing projections. One possible solution would be to insert new measures in an existing projection. This, however, requires a deterministic projection function and the quality may decrease as the projection is computed based on the old patterns only. Another option would be to recompute the projection every time a new pattern is added. While this is a sound approach in

theory, it will be challenging to communicate the changes between the old and the new projection effectively to the user, especially if it changes significantly. One lesson we learned is the role of familiar visualizations to enable the learning of new techniques. Here, the calendar view was the anchoring point for the user to understand the rest of the system featuring visualizations which have not been used before.

5.2. Outlook

From a scientific perspective, the combined analysis of graph and time series data still poses a vast amount of unsolved challenges. In this thesis the analysis of a set of univariate time-series data that is assigned to a corresponding set of graph nodes was presented. Two challenges can be directly derived from this. The first is about multi-variate temporal data, the second is about mixed data analysis.

Multi-variate temporal data is highly relevant in many applications. In earth observation, for example, sensor typically measure a variety of individual quantities such as temperature, humidity, downfall and the like. Finding correlations between different types of variables is challenging not only because adequate definitions of similarity must be put up. Several approaches in that direction have been made (e.g. in the TimeSeriesPaths visualization tool by Bernard et al. [BWS*12]), but not to an exhaustive extent.

In analogy to that, the analysis of mixed data in combination with temporal network data also asks for new analysis tools. In contrast to additional temporal data, the data comprises a variety of categorical or ordinal values. While coping with multi-variate feature spaces is a problem on its own, the combined analysis of these two feature spaces implies a particular challenge. Such combinations occur in a variety of application domains such as medical science, industrial manufacturing, and environmental services, confronting domain experts with the question how to gain insight in an effective and efficient way. Again, the answer(s) to that question have not been thoroughly discussed yet and will surely gain interest in both the scientific community, but also by practitioners.

Apart from the data complexity, the amount of data begins to become a challenge, since data volumes become larger and larger. Along with increasing *volume* comes an increase of *variety* and at higher processing *velocity*. These three words are often used to denote *Big Data*, sometimes complemented by *veracity*, indicating different levels of consistency or trustworthiness. This combination brings conventional hardware and software architecture to its limits, asking for new setups. Under the headline of *Big Data Analytics*, large hardware infrastructures have been set up to tackle scalability problems. Based on the map-reduce paradigm, software is distributed to where the data is rather than the other way around. Notably, large parts of developed software was and still is made available for public use under an open-source license.

Enterprises that publish the internals of a software project do that for a good reason. Requirements for data analysis have been going up, leading to highly sophisticated tools. Developing and maintaining such complex software is becoming increasingly costly. As a consequence, re-using software has turned from a nice-to-have-feature into an absolute must. Thus, recent software tends to be generic rather than a custom-tailored, individual product. Only in a second step, this

common product is adjusted to a specific business case through add-ons, plugins, etc. This opens up possibilities for third-party users, which makes this open access principle more attractive than ever.

A logical consequence is that software is longer created by a single individual or enterprise, but comprises a conglomerate of many intertwined projects and libraries. Integration and coupling of involved systems requires very careful design and suitable architectures. One strategy that emerged recently and gained strong interest since then is loose coupling based on *reactive systems*. Instead of combining all software parts into a fully integrated product, components act as more or less self-contained actors in a network. Communication is done through message buses rather than direct method calls. This obviously comes with performance penalties, but provides enormous flexibility. An additional benefit is that such software can be distributed to different machines and communicate across physical boundaries. Parts can be added and removed (sometimes even at runtime) without breaking the environment. Many of the services the software tools offer can be easily provided on the web through message adapters.

This is of great value: we are convinced that the trend towards web-based visualizations will continue. Many users of visualization and analysis tools are not overly computer-savvy. Web applications lowers the usage barrier as no software must be downloaded and installed. Also, software security is hardly an issue on web-based imagery. Closely linked is the usage on mobile – in particular handheld – devices. In our opinion, visualizations on such devices can be very useful for information presentation. One reason is that interaction is mostly limited to “touch”. Even though screen resolution increases continuously, the physical size is rather limited. As the finger covers the display while pressing, it can be rather difficult to hit small interaction elements. This could change with the ability to interact and control software through voice commands, but it may take a few more years until that works reliable enough.

A. Publications

Relevant Publications

The thesis is partially based on the following publications:

2015

- EXPLORATIVE ANALYSIS OF 2D COLOR MAPS
Steiger, Martin; Bernard, Jürgen; Hutter, Marco; Thum, Simon; Mittelstädt, Sebastian; Keim, Daniel; Kohlhammer, Jörn [[SBM*15](#)]

2014

- EXPLORING SIMULATION IN SENSOR NETWORK MODELS
Steiger, Martin; Hutter, Marco; Schader, Philipp; Kohlhammer, Jörn and Kuijper, Arjan [[SHS*14](#)].
- VISUAL ANALYSIS OF TIME-SERIES SIMILARITIES FOR ANOMALY DETECTION IN SENSOR NETWORKS
Steiger, Martin; Bernard, Jürgen; Mittelstädt, Sebastian; Lücke-Tieke, Hendrik; Keim, Daniel; May, Thorsten; Kohlhammer, Jörn [[SBM*14](#)]
- A SURVEY OF DIRECTION-PRESERVING LAYOUT STRATEGIES
Steiger, Martin; Bernard, Jürgen; May, Thorsten; Kohlhammer, Jörn) [[SBMK14](#)].
- DETERMINISTIC LOCAL LAYOUTS THROUGH HIGH-DIMENSIONAL LAYOUT STITCHING
Steiger, Martin; Lücke-Tieke, Hendrik; May, Thorsten; Kuijper, Arjan; Kohlhammer, Jörn [[SLTM*14](#)].
- INFORMATION VISUALIZATION AND POLICY MODELING
Nazemi, Kawa; Steiger, Martin; Burkhardt, Dirk; Kohlhammer, Jörn [[NSBK14](#)].

2013

- STABLE INCREMENTAL LAYOUTS FOR DYNAMIC GRAPH VISUALIZATIONS
Steiger, Martin; May, Thorsten; Kohlhammer, Jörn [[SMK13](#)], [[SMK14](#)] (Journal article).

- USING LAYOUT STITCHING TO CREATE DETERMINISTIC LOCAL GRAPH LAYOUTS
Steiger, Martin; Lücke-Tieke, Hendrik; May, Thorsten; Kuijper, Arjan; Kohlhammer, Jörn [[SLTM*13](#)].
- SMART GRID MONITORING THROUGH VISUAL ANALYSIS
Steiger, Martin; May, Thorsten; Davey, James; Kohlhammer, Jörn [[SMDK13a](#)].
- VISUAL ANALYSIS OF EXPERT SYSTEMS FOR SMART GRID MONITORING
Steiger, Martin; May, Thorsten; Davey, James; Kohlhammer, Jörn [[SMDK13b](#)].

2012

- USING SIGNPOSTS FOR NAVIGATION IN LARGE GRAPHS
May, Thorsten; Steiger, Martin; Davey, James; Kohlhammer, Jörn [[MSDK12](#)]

Other Publications

2015

- VISUAL ANALYSIS OF RELATIONS IN ATTRIBUTED TIME-SERIES DATA
Steiger, Martin; Bernard, Jürgen; Schader, Philipp; Kohlhammer, Jörn [[SBSK15](#)]
- A SURVEY AND TASK-BASED QUALITY ASSESSMENT OF STATIC 2D COLOR MAPS
Bernard, Jürgen; Steiger, Martin; Mittelstädt, Sebastian; Thum, Simon; Keim, Daniel; Kohlhammer, Jörn [[BSM*15](#)]

2014

- INTERACTIVE MULTI-CRITERIA OPTIMIZATION OF 2D COLOR MAPS
Hutter, Marco; Steiger, Martin; Bernard, Jürgen; Zurlöf, Corinna; Kohlhammer, Jörn
[Poster at VMV Conference, 2015]
- VISUAL-INTERACTIVE EXPLORATION OF INTERESTING MULTIVARIATE RELATIONS IN MIXED RESEARCH DATA SETS
Bernard, Jürgen; Steiger, Martin; Widmer, Sven; Lücke-Tieke, Hendrik; May, Thorsten; Kohlhammer, Jörn [[BSW*14](#)]
- REVISITING PERCEPTUALLY OPTIMIZED COLOR MAPPING FOR HIGH-DIMENSIONAL DATA ANALYSIS
Mittelstädt, Sebastian; Bernard, Jürgen; Schreck, Tobias; Steiger, Martin; Kohlhammer, Jörn; Keim, Daniel [[MBS*14](#)]

2013

- VISUAL STATISTICS COCKPITS FOR INFORMATION GATHERING IN THE POLICY-MAKING PROCESS
Burkhardt, Dirk; Nazemi, Kawa; Stab, Christian; Steiger, Martin; Kuijper, Arjan; Kohlhammer, Jörn [[BNS*13](#)]
- VISUALIZING UNCERTAIN UNDERGROUND INFORMATION FOR URBAN MANAGEMENT
Steiger, Martin; Krämer, Michel; Ruppert, Tobias; Kohlhammer, Jörn [[SKRK13](#)]

2012

- INTERACTIVE EXPLORATION SYSTEM: A USER-CENTERED INTERACTION APPROACH IN SEMANTICS VISUALIZATIONS
Burkhardt, Dirk; Stab, Christian; Steiger, Martin; Breyer, Matthias; Nazemi, Kawa [[BSS*12](#)]

2011

- TACKLING UNCERTAINTY IN COMBINED VISUALIZATIONS OF UNDERGROUND INFORMATION AND 3D CITY MODELS
Krämer, Michel; Steiger, Martin; Ruppert, Tobias; Kohlhammer, Jörn [[KDRK11](#)]

B. Supervising Activities

The following list summarizes the student bachelor, diploma and master thesis supervised by the author. The results of these works were partially used as an input into the thesis.

Diploma and Master Theses

- Xin Zhou – Implementierung einer Stoffsimulation und deren Parallelisierung
- Daniel Tillner – Metro Graph Layouts für Stromnetze

Bachelor Theses

- Bernd Klemm – Transport und Visualisierung von 3D-Daten in Echtzeit
- Philipp Schader – Assistiertes Energiemanagement durch Visuelle Analyse

Internships

- Dennis Klugmann – Working with Linux and LaTeX, updating Wiki-Content
- Corinna Zurloh – Interactive Multi-Criteria Optimization of 2D Color Maps
- Alexander Distergoft – Programmieren eines grafischen Systems

Bibliography

- [AAD*10] ANDRIENKO G., ANDRIENKO N., DEMSAR U., DRANSCH D., DYKES J., FABRIKANT S. I., JERN M., KRAAK M.-J., SCHUMANN H., TOMINSKI C.: Space, time and visual analytics. *Int. J. Geogr. Inf. Sci.* 24, 10 (Oct. 2010), 1577–1600. [92](#)
- [AAS*12] ANDRIENKO N., ANDRIENKO G., STANGE H., LIEBIG T., HECKER D.: Visual analytics for understanding spatial situations from episodic movement data. *KI-Künstliche Intelligenz* 26, 3 (2012), 241–251. [137](#)
- [Add12] ADDITION, LDA.: EPANET Java, 2012. [160](#)
- [AHB87] ARUN K. S., HUANG T. S., BLOSTEIN S. D.: Least-Squares Fitting of Two 3D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-9*, 5 (sept. 1987), 698–700. [70](#), [71](#)
- [Alb10] ALBRECHT M.: Color blindness. *Nature Methods* 7 (2010), 775. [135](#)
- [AM00] AVELAR S., MÜLLER M.: Generating topologically correct schematic maps. In *Proceedings of the 9th International Symposium on Spatial Data Handling* (2000), pp. 4–28. [102](#)
- [AMA07] ARCHAMBAULT D., MUNZNER T., AUBER D.: TopoLayout: Multilevel Graph Layout by Topological Features. *Visualization and Computer Graphics, IEEE Transactions on* 13, 2 (march-april 2007), 305–317. [38](#)
- [AME11] AFZAL S., MACIEJEWSKI R., EBERT D.: Visual analytics decision support environment for epidemic modeling and response evaluation. In *VAST* (Oct 2011), pp. 191–200. [152](#)
- [AMST11] AIGNER W., MIKSCH S., SCHUMANN H., TOMINSKI C.: *Visualization of Time-Oriented Data*. Springer, London, UK, 2011. [92](#)
- [And95] ANDREWS K.: Case study. visualising cyberspace: information visualisation in the harmony internet browser. In *Information Visualization, 1995. Proceedings.* (Oct 1995), pp. 97–104. [24](#)
- [AP13] ARCHAMBAULT D., PURCHASE H.: Mental map preservation helps user orientation in dynamic graphs. In *Graph Drawing*, Didimo W., Patrignani M., (Eds.), vol. 7704 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 475–486. [39](#)

- [APP11] ARCHAMBAULT D., PURCHASE H., PINAUD B.: Animation, Small Multiples, and the Effect of Mental Map Preservation in Dynamic Graphs. *Visualization and Computer Graphics, IEEE Transactions on* 17, 4 (april 2011), 539–552. [40](#)
- [AS05] AMAR R., STASKO J.: Knowledge precepts for design and evaluation of information visualizations. *Visualization and Computer Graphics, IEEE Transactions on* 11, 4 (2005), 432–442. [11](#)
- [Asi85] ASIMOV D.: The grand tour: A tool for viewing multidimensional data. *SIAM J. Sci. Stat. Comput.* 6, 1 (Jan. 1985), 128–143. [14](#)
- [AvHK06] ABELLO J., VAN HAM F., KRISHNAN N.: ASK-GraphView: A Large Scale Graph Visualization System. *IEEE Transactions on Visualization and Computer Graphics* 12 (September 2006), 669–676. [37](#)
- [BAF*13] BÖGL M., AIGNER W., FILZMOSE P., LAMMARSCH T., MIKSCH S., RIND A.: Visual analytics for model selection in time series analysis. *IEEE Trans. Vis. Comput. Graph.* 19, 12 (2013), 2237–2246. [92](#)
- [BBDW14] BECK F., BURCH M., DIEHL S., WEISKOPF D.: The state of the art in visualizing dynamic graphs. In *EuroVis - STARs* (2014), Eurographics Association, pp. 83–103. [38](#)
- [BBDZ08] BÖTTGER J., BRANDES U., DEUSSEN O., ZIEZOLD H.: Map warping for the annotation of metro maps. *Computer Graphics and Applications, IEEE* 28, 5 (2008), 56–65. [105](#)
- [BC94] BERNDT D. J., CLIFFORD J.: Using dynamic time warping to find patterns in time series. In *KDD workshop* (1994), vol. 10/16, Seattle, WA, pp. 359–370. [123](#)
- [BD07] BALZER M., DEUSSEN O.: Level-of-detail visualization of clustered graph layouts. In *Visualization, 2007. APVIS '07. 2007 6th International Asia-Pacific Symposium on* (2007), pp. 133–140. [37](#)
- [Ber67] BERTIN J.: *Sémiologie graphique: les diagrammes, les réseaux, les cartes*. Mouton, 1967. [7](#)
- [Ber02] BERKHIN P.: Survey of clustering data mining techniques. *Techniques* 10, c (2002), 1–56. [52](#)
- [BGW03] BRANDES U., GAERTLER M., WAGNER D.: Experiments on graph clustering algorithms. In *Algorithms - ESA 2003*, Battista G., Zwick U., (Eds.), vol. 2832 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003, pp. 568–579. [36](#)

-
- [BH94] BEDERSON B. B., HOLLAN J. D.: Pad++: a zooming graphical interface for exploring alternate interface physics. In *Proceedings of the 7th annual ACM symposium on User interface software and technology* (New York, NY, USA, 1994), UIST '94, ACM, pp. 17–26. [22](#)
- [Bie06] BIEMANN C.: Chinese Whispers: an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing* (Stroudsburg, PA, USA, 2006), TextGraphs-1, Association for Computational Linguistics, pp. 73–80. [76](#)
- [BKSW07] BEKOS M. A., KAUFMANN M., SYMVONIS A., WOLFF A.: Boundary labeling: Models and efficient algorithms for rectangular maps. *Computational Geometry* 36, 3 (2007), 215–236. [104](#)
- [BL07] BROWN M., LOWE D. G.: Automatic Panoramic Image Stitching using Invariant Features. *International Journal of Computer Vision* 74 (2007), 59–73. [70](#)
- [BLR00] BARKOWSKY T., LATECKI L. J., RICHTER K.-F.: Schematizing maps: Simplification of geographic shape by discrete curve evolution. In *Spatial Cognition II*, vol. 1849 of *Lecture Notes in Computer Science*. Springer, Heidelberg, 2000, pp. 41–53. [102](#)
- [BM10] BRUCKNER S., MÖLLER T.: Result-driven exploration of simulation parameter spaces for visual effects design. *Visualization and Computer Graphics, IEEE Transactions on* 16, 6 (2010), 1468–1476. [153](#)
- [BM12] BRANDES U., MADER M.: A quantitative comparison of stress-minimization approaches for offline dynamic graph drawing. In *Graph Drawing*, Kreveld M., Speckmann B., (Eds.), vol. 7034 of *LNCS*. Springer Berlin Heidelberg, 2012, pp. 99–110. [39](#)
- [BM13] BREHMER M., MUNZNER T.: A multi-level typology of abstract visualization tasks. *Visualization and Computer Graphics, IEEE Transactions on* 19, 12 (Dec 2013), 2376–2385. [9](#), [27](#)
- [BNS*13] BURKHARDT D., NAZEMI K., STAB C., STEIGER M., KUIJPER A., KOHLHAMMER J.: Visual statistics cockpits for information gathering in the policy-making process. In *Advances in Visual Computing*, Bebis G., Boyle R., Parvin B., Koracin D., Li B., Porikli F., Zordan V., Klosowski J., Coquillart S., Luo X., Chen M., Gotz D., (Eds.), vol. 8034 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 86–97. [173](#)
- [Bos08] BOSTRIDGE M.: *Florence Nightingale: the making of an icon*. Macmillan, 2008. [7](#)
-

- [BP09] BRANDES U., PICH C.: An experimental study on distance-based graph drawing. In *Graph Drawing*, vol. 5417 of *Lecture Notes in Computer Science*. Springer, 2009, pp. 218–229. [37](#), [123](#)
- [BPFG11] BERGER W., PIRINGER H., FILZMOSER P., GRÖLLER E.: Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. In *Computer Graphics Forum* (2011), vol. 30/3, pp. 911–920. [153](#)
- [BPW*93] BRODLIE K., POON A., WRIGHT H., BRANKIN L., BANECKI G., GAY A.: GRASPARC—a problem solving environment integrating computation and visualization. In *IEEE Conference on Visualization* (Oct 1993), pp. 102–109. [153](#)
- [BR03] BAUDISCH P., ROSENHOLTZ R.: Halo: a technique for visualizing off-screen objects. In *Proc. of the 21th International Conference on Human Factors in Computing Systems (CHI)* (New York, NY, USA, 2003), CHI '03, ACM, pp. 481–488. [41](#)
- [Bra01] BRANKE J.: Dynamic graph drawing. In *Drawing Graphs*, Kaufmann M., Wagner D., (Eds.), vol. 2025 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 228–246. [40](#)
- [BRG*12] BERNARD J., RUPPERT T., GOROLL O., MAY T., KOHLHAMMER J.: Visual-interactive preprocessing of time series data. In *SIGRAD* (2012), pp. 39–48. [92](#)
- [BRS*12a] BERNARD J., RUPPERT T., SCHERER M., KOHLHAMMER J., SCHRECK T.: Content-based layouts for exploratory metadata search in scientific research data. In *Proc. of the Joint Conference on Digital Libraries* (2012), JCDL, ACM, pp. 139–148. [94](#)
- [BRS*12b] BERNARD J., RUPPERT T., SCHERER M., KOHLHAMMER J., SCHRECK T.: Content-based layouts for exploratory metadata search in scientific research data. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries* (New York, NY, USA, 2012), JCDL '12, ACM, pp. 139–148. [137](#)
- [BRT95] BERGMAN L., ROGOWITZ B., TREINISH L.: A rule-based tool for assisting colormap selection. In *Visualization, 1995. Visualization '95. Proceedings., IEEE Conference on* (Oct 1995), pp. 118–125, 444. [137](#)
- [BSM*15] BERNARD J., STEIGER M., MITTELSTÄDT S., THUM S., KEIM D. A., KOHLHAMMER J.: A survey and task-based quality assessment of static 2D color maps. In *To be published at the Conference on Visualization and Data Analysis* (2015). [135](#), [137](#), [139](#), [172](#)
- [BSS*12] BURKHARDT D., STAB C., STEIGER M., BREYER M., NAZEMI K.: Interactive exploration system: A user-centered interaction approach in semantics

- visualizations. In *Cyberworlds (CW), 2012 International Conference on* (Sept 2012), pp. 261–267. [173](#)
- [BSW*14] BERNARD J., STEIGER M., WIDMER S., LÜCKE-TIEKE H., MAY T., KOHLHAMMER J.: Visual-interactive Exploration of Interesting Multivariate Relations in Mixed Research Data Sets. *Comp. Graph. Forum* 33, 3 (2014), 291–300. [137](#), [172](#)
- [BTK11] BERTINI E., TATU A., KEIM D. A.: Quality metrics in high-dimensional data visualization: An overview and systematization. *IEEE Symposium on Information Visualization (InfoVis)* 17, 12 (Dec. 2011), pages 2203–2212. [123](#)
- [BvLBS09] BERNARD J., VON LANDESBERGER T., BREMM S., SCHRECK T.: Micro-macro views for visual trajectory cluster analysis. In *IEEE Symposium on Visualization* (2009). [126](#)
- [BvLBS11] BREMM S., VON LANDESBERGER T., BERNARD J., SCHRECK T.: Assisted descriptor selection based on visual comparative data analysis. In *Computer Graphics Forum* (2011), vol. 30/3, Wiley Online Library, pp. 891–900. [127](#), [137](#), [149](#)
- [BWK*13] BERNARD J., WILHELM N., KRÜGER B., MAY T., SCHRECK T., KOHLHAMMER J.: Motionexplorer: Exploratory search in human motion capture data based on hierarchical aggregation. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2257–2266. [123](#), [137](#)
- [BWS*12] BERNARD J., WILHELM N., SCHERER M., MAY T., SCHRECK T.: Time-seriespaths: Projection-based explorative analysis of multivariate time series data. *Journal of WSCG* 20, 2 (2012), 97–106. [95](#), [168](#)
- [CBB*12] COMPTON M., BARNAGHI P., BERMUDEZ L., GARCÍA-CASTRO R., CORCHO O., COX S., GRAYBEAL J., HAUSWIRTH M., HENSON C., HERZOG A., HUANG V., JANOWICZ K., KELSEY W. D., PHUOC D. L., LEFORT L., LEGGIERI M., NEUHAUS H., NIKOLOV A., PAGE K., PASSANT A., SHETH A., TAYLOR K.: The {SSN} ontology of the {W3C} semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web* 17, 0 (2012), 25 – 32. [92](#)
- [CCF97] CARPENDALE M., COWPERTHWAIT D., FRACCHIA F.: Extending distortion viewing from 2d to 3d. *Computer Graphics and Applications* 17, 4 (1997), 42–51. [105](#)
- [CdBvD*01] CABELLO S., DE BERG M., VAN DIJK S., VAN KREVELD M., STRIJK T.: Schematization of road networks. In *Proceedings of the Seventeenth Annual Symposium on Computational Geometry* (New York, NY, USA, 2001), SCG '01, ACM, pp. 33–39. [102](#)

- [CdBvK05] CABELLO S., DE BERG M., VAN KREVELD M.: Schematization of networks. *Computational Geometry* 30, 3 (2005), 223–238. [97](#)
- [CEH*09] CHEN M., EBERT D., HAGEN H., LARAMEE R., VAN LIERE R., MA K.-L., RIBARSKY W., SCHEUERMANN G., SILVER D.: Data, information, and knowledge in visualization. *Computer Graphics and Applications, IEEE* 29, 1 (jan.-feb. 2009), 12 –19. [94](#)
- [CHL*09] COMPTON M., HENSON C., LEFORT L., NEUHAUS H., SHETH A.: A survey of the semantic specification of sensors. *Proc. Semantic Sensor Networks* 17 (2009). [92](#)
- [CK03] CABELLO S., KREVELD M.: Approximation algorithms for aligning points. *Algorithmica* 37, 3 (2003), 211–232. [102](#)
- [CKB08] COCKBURN A., KARLSON A., BEDERSON B. B.: A review of overview+detail, zooming, and focus+context interfaces. *ACM Comput. Surv.* 41, 1 (Jan. 2008), 2:1–2:31. [24](#), [39](#), [96](#)
- [CLWM11] CRNOVRSANIN T., LIAO I., WU Y., MA K.-L.: Visual recommendations for network navigation. In *IEEE Symposium on Visualization (EuroVis 2011)* (2011), Hauser H., Pfister H., van Wijk J. J., (Eds.), vol. 30, Blackwell Publishing Ltd., pp. 1081–1090. [41](#)
- [CMS99a] CARD S. K., MACKINLAY J. D., SHNEIDERMAN B. (Eds.): *Readings in Information Visualization – Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999. [8](#), [24](#), [94](#)
- [CMS99b] CARD S. K., MACKINLAY J. D., SHNEIDERMAN B.: *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999. [27](#)
- [CNM83] CARD S. K., NEWELL A., MORAN T. P.: *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1983. [8](#)
- [CPC09] COLLINS C., PENN G., CARPENDALE S.: Bubble sets: Revealing set relations with isocontours over existing visualizations. *Visualization and Computer Graphics, IEEE Transactions on* 15, 6 (2009), 1009–1016. [133](#)
- [CPF09] CLARK A., PAVLOVSKI C., FRY J.: Transformation of energy systems: The control room of the future. In *Electrical Power Energy Conference (EPEC), 2009 IEEE* (oct. 2009), pp. 1 –6. [89](#), [90](#), [94](#), [95](#)
- [CRM91] CARD S. K., ROBERTSON G. G., MACKINLAY J. D.: The information visualizer, an information workspace. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1991), CHI '91, ACM, pp. 181–186. [8](#)

-
- [CSP*06] CARD S. K., SUH B., PENDLETON B. A., HEER B., BODNAR J. W.: Time tree: Exploring time changing hierarchies. In *IEEE VAST* (2006), pp. 3–10. [13](#)
- [CYG04] CAMGÖZ N., YENER C., GÜVENC D.: Effects of hue, saturation, and brightness: Part 2: Attention. *Color Research & Application* 29, 1 (2004), 20–28. [140](#)
- [DGK01] DIEHL S., GÖRG C., KERREN A.: Preserving the mental map using foresighted layout. In *Proceedings of the 3rd Joint Eurographics - IEEE TCVG Conference on Visualization* (Aire-la-Ville, Switzerland, Switzerland, 2001), EGVISYMM'01, Eurographics Association, pp. 175–184. [40](#)
- [DMK05] DYKES J., MACEACHREN A. M., KRAAK M.-J.: *Exploring Geovisualization*. International Cartographic Association. Elsevier, 2005. [14](#)
- [DMS*08] DWYER T., MARRIOTT K., SCHREIBER F., STUCKEY P., WOODWARD M., WYBROW M.: Exploration of Networks using overview+detail with Constraint-based cooperative layout. *IEEE Transactions on Visualization and Computer Graphics* 14, 6 (Nov. 2008), 1293–1300. [39](#), [41](#)
- [DMW09] DWYER T., MARRIOTT K., WYBROW M.: Topology Preserving Constrained Graph Layout. In *Graph Drawing*, Tollis I., Patrignani M., (Eds.), vol. 5417 of *LNCS*. Springer, 2009, pp. 230–241. [97](#)
- [dNE08] DO NASCIMENTO H. A., EADES P.: User hints for map labeling. *Journal of Visual Languages & Computing* 19, 1 (2008), 39–74. Spatial and Image-based Information Systems. [104](#)
- [DTS*08] DING H., TRAJCEVSKI G., SCHEUERMANN P., WANG X., KEOGH E.: Querying and mining of time series data: Experimental comparison of representations and distance measures. *Proc. VLDB Endow.* 1, 2 (Aug. 2008), 1542–1552. [92](#)
- [Ead84] EADES P.: A heuristics for graph drawing. *Congressus numerantium* 42 (1984), 146–160. [37](#)
- [EDF08] ELMQVIST N., DRAGICEVIC P., FEKETE J.: Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *Visualization and Computer Graphics, IEEE Transactions on* 14, 6 (2008), 1539–1148. [15](#)
- [EFK01] EIGLSPERGER M., FEKETE S. P., KLAU G. W.: Orthogonal graph drawing. In *Drawing Graphs*, Kaufmann M., Wagner D., (Eds.), vol. 2025 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 121–171. [100](#)
- [EHK*04] ERTEN C., HARDING P. J., KOBOUROV S. G., WAMPLER K., YEE G.: GraphAEL: Graph Animations with Evolving Layouts. In *Graph Drawing*, Liotta G., (Ed.), vol. 2912 of *LNCS*. Springer, 2004, pp. 98–110. [39](#)
-

- [EKM11] EROL-KANTARCI M., MOUFTAH H.: Wireless sensor networks for cost-efficient residential energy management in the smart grid. *Smart Grid, IEEE Transactions on* 2, 2 (2011), 314–325. [111](#)
- [ELMS91] EADES P., LAI W., MISUE K., SUGIYAMA K.: Preserving the mental map of a diagram. *Proceedings of COMPUGRAPHICS 91*, 9 (1991), 24–33. [33](#), [39](#)
- [FD10] FRISCH M., DACHSELT R.: Off-screen visualization techniques for class diagrams. In *Proc. of the 5th International Symposium on Software visualization* (New York, NY, USA, 2010), SOFTVIS '10, ACM, pp. 163–172. [41](#)
- [FHN*13] FINK M., HAVERKORT H., NÖLLENBURG M., ROBERTS M., SCHUHMANN J., WOLFF A.: Drawing metro maps using bézier curves. In *Graph Drawing*, Didimo W., Patrignani M., (Eds.), vol. 7704 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 463–474. [101](#)
- [Fis10] FISHER D.: Animation for visualization: opportunities and drawbacks. *Beautiful Visualization: Looking at Data Through the Eyes of Experts 19* (2010), 329–352. [23](#)
- [FK96] FORMELLA A., KELLER J.: Generalized fisheye views of graphs. In *Graph Drawing*, Brandenburg F. J., (Ed.), vol. 1027 of *LNCS*. Springer, 1996, pp. 242–253. [38](#)
- [FP99] FEKETE J.-D., PLAISANT C.: Excentric labeling: Dynamic neighborhood labeling for data visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1999), CHI '99, ACM, pp. 512–519. [104](#), [108](#)
- [FR91] FRUCHTERMAN T. M. J., REINGOLD E. M.: Graph drawing by force-directed placement. *Software: Practice and Experience* 21, 11 (1991), 1129–1164. [37](#), [58](#)
- [FT04] FRISHMAN Y., TAL A.: Dynamic Drawing of Clustered Graphs. In *IEEE Symposium on Information Visualization (InfoVis '04)* (2004), pp. 191–198. [38](#), [39](#), [40](#)
- [FT08] FRISHMAN Y., TAL A.: Online Dynamic Graph Drawing. *IEEE Transactions on Visualization and Computer Graphics* 14, 4 (july-aug. 2008), 727 –740. [39](#), [40](#)
- [Fu11] FU T.-C.: A review on time series data mining. *Eng. Appl. Artif. Intell.* 24, 1 (Feb. 2011), 164–181. [92](#)
- [Fur86] FURNAS G. W.: Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1986), CHI '86, ACM, pp. 16–23. [7](#), [23](#), [40](#), [43](#), [46](#)

-
- [FvWSN08] FEKETE J.-D., VAN WIJK J. J., STASKO J. T., NORTH C.: The value of information visualization. In *Information Visualization*, Kerren A., Stasko J. T., Fekete J.-D., North C., (Eds.), vol. 4950 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 1–18. [24](#)
- [FWR99] FUA Y.-H., WARD M. O., RUNDENSTEINER E. A.: Hierarchical parallel coordinates for exploration of large datasets. In *Proceedings of the Conference on Visualization '99: Celebrating Ten Years* (Los Alamitos, CA, USA, 1999), VIS '99, IEEE Computer Society Press, pp. 43–50. [16](#)
- [GB94] GARLAND K., BECK H. C.: *Mr Beck's Underground Map*. Capital Transport, 1994. [7](#), [97](#)
- [GBPD05] GÖRG C., BIRKE P., POHL M., DIEHL S.: Dynamic Graph Drawing of Sequences of Orthogonal and Hierarchical Graphs. In *Graph Drawing*, Pach J., (Ed.), vol. 3383 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 228–238. [39](#)
- [GCML06] GUO D., CHEN J., MACEACHREN A., LIAO K.: A visualization system for space-time and multivariate patterns (vis-stamp). *Visualization and Computer Graphics, IEEE Transactions on* 12, 6 (Nov 2006), 1461–1474. [137](#)
- [GCV*07] GOH K.-I., CUSICK M. E., VALLE D., CHILDS B., VIDAL M., BARABÁSI A.-L.: The Human Disease Network. *Proc. of the National Academy of Sciences USA* 104, 21 (2007), 8685–8690. [29](#), [55](#), [76](#)
- [GFC05] GHONIEM M., FEKETE J.-D., CASTAGLIOLA P.: On the readability of graphs using node-link and matrix-based representations: A controlled experiment and statistical analysis. *Information Visualization* 4, 2 (2005), 114–135. [35](#)
- [GGK01] GAJER P., GOODRICH M., KOBOUROV S.: A fast multi-dimensional approach to force-directed layouts of large graphs. In *Graph Drawing*, vol. 1984 of *LNCS*. Springer, 2001, pp. 211–221. [37](#), [38](#), [80](#)
- [GGMZ05] GUO D., GAHEGAN M., MACEACHREN A. M., ZHOU B.: Multivariate analysis and geovisualization with an integrated geographic knowledge discovery approach. *Cartography and Geographic IS* 32, 2 (2005), 113–132. [137](#), [149](#)
- [GNBP11] GASTEIGER R., NEUGEBAUER M., BEUING O., PREIM B.: The FLOWLENS: A focus-and-context visualization approach for exploration of blood flow in cerebral aneurysms. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (dec. 2011), 2183–2192. [44](#)
- [GNRM08] GARG S., NAM J., RAMAKRISHNAN I., MUELLER K.: Model-driven visual analytics. In *Visual Analytics Science and Technology, 2008. VAST '08. IEEE Symposium on* (oct. 2008), pp. 19–26. [94](#)
-

- [GP13] GRISCOM W., PALMER S.: Violins are green, pianos are blue: Cross-modal sound-to-sight associations with timbre in synesthetes & non-synesthetes. *Journal of Vision* 13, 9 (2013), 1169–1169. [137](#)
- [GRE11] GHANI S., RICHE N. H., ELMQVIST N.: Dynamic insets for context-aware graph navigation. *Computer Graphics Forum* 30, 3 (2011), 861–870. [41](#), [45](#)
- [GSGC08] GILSON O., SILVA N., GRANT P., CHEN M.: From web data to visualization via ontology mapping. *Computer Graphics Forum* 27, 3 (2008), 959–966. [94](#)
- [HAB09] HAY S., AULT G., BELL K.: Control room scenarios on active distribution networks: Early results and next steps. In *Universities Power Engineering Conference (UPEC), 2009 Proceedings of the 44th International* (sept. 2009), pp. 1–5. [89](#), [91](#)
- [HABM08] HAY S., AULT G., BELL K., McDONALD J.: System operator interfaces to active network management schemes in future distribution networks. In *Universities Power Engineering Conference, 2008. UPEC 2008. 43rd International* (sept. 2008), pp. 1–4. [92](#)
- [HB03a] HARROWER M., BREWER C. A.: ColorBrewer.org: An online tool for selecting colour schemes for maps. *The Cartographic Journal* 40, 1 (2003), 27–37. [136](#)
- [HB03b] HARROWER M., BREWER C. A.: Colorbrewer.org: An online tool for selecting colour schemes for maps. *The Cartographic Journal* 40, 1 (2003), 27–37. [156](#)
- [HC04] HEER J., CARD S. K.: Doitrees revisited: Scalable, space-constrained visualization of hierarchical data. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (New York, NY, USA, 2004), AVI '04, ACM, pp. 421–424. [46](#), [47](#)
- [HDKS05] HAO M., DAYAL U., KEIM D., SCHRECK T.: Importance-driven visualization layouts for large time series data. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on* (2005), pp. 203–210. [13](#), [19](#)
- [HE98] HUANG M. L., EADES P.: A Fully Animated Interactive System for Clustering and Navigating Huge Graphs. In *Proceedings of the 6th International Symposium on Graph Drawing* (London, UK, 1998), GD '98, Springer-Verlag, pp. 374–383. [38](#), [62](#)
- [Hea96] HEALEY C. G.: Choosing effective colours for data visualization. In *Visualization* (1996), IEEE, pp. 263–270. [136](#)
- [Hea99] HEARST M. A.: User interfaces and visualization. *Modern information retrieval* (1999), 257–323. [21](#), [23](#)

-
- [HEL05] HUANG X., EADES P., LAI W.: A framework of filtering, clustering and dynamic layout graphs for visualization. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science - Volume 38* (Darlinghurst, Australia, Australia, 2005), ACSC '05, Australian Computer Society, Inc., pp. 87–96. [38](#)
- [HEW98] HUANG M. L., EADES P., WANG J.: Online Animated Visualization of Huge Graphs using a Modified Spring Algorithm. *Journal of Visual Languages & Computing* 9, 6 (1998), 623–645. [38](#), [48](#)
- [HFM07] HENRY N., FEKETE J.-D., MCGUFFIN M.: NodeTriX: a Hybrid Visualization of Social Networks. *Visualization and Computer Graphics, IEEE Transactions on* 13, 6 (nov.-dec. 2007), 1302–1309. [37](#)
- [HHN00] HAVRE S., HETZLER B., NOWELL L.: Themeriver: visualizing theme changes over time. In *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on* (2000), pp. 115–123. [13](#)
- [Him98] HIMBERG J.: Enhancing the som based data visualization by linking different data projections. In *Proceedings of 1st International Symposium IDEAL* (1998), vol. 98, pp. 427–434. [137](#)
- [HIvWF11] HOLTEN D., ISENBERG P., VAN WIJK J., FEKETE J.: An extended evaluation of the readability of tapered, animated, and textured directed-edge representations in node-link graphs. In *Pacific Visualization Symposium (PacificVis), 2011 IEEE* (March 2011), pp. 195–202. [161](#)
- [HK01] HAREL D., KOREN Y.: A fast multi-scale method for drawing large graphs. In *Graph Drawing*, Marks J., (Ed.), vol. 1984 of *LNCS*. Springer, 2001, pp. 183–196. [38](#)
- [HK02] HAREL D., KOREN Y.: Graph drawing by high-dimensional embedding. In *Graph Drawing*, Goodrich M., Kobourov S., (Eds.), vol. 2528 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2002, pp. 207–219. [37](#)
- [HMdN06] HONG S.-H., MERRICK D., DO NASCIMENTO H. A.: Automatic visualisation of metro maps. *Journal of Visual Languages & Computing* 17, 3 (2006), 203 – 224. [98](#), [100](#), [102](#)
- [HMM00] HERMAN I., MELANCON G., MARSHALL M.: Graph visualization and navigation in information visualization: A survey. *Visualization and Computer Graphics, IEEE Transactions on* 6, 1 (2000), 24–43. [34](#)
- [HMN05] HONG S.-H., MERRICK D., NASCIMENTO H.: The metro map layout problem. In *Graph Drawing*, Pach J., (Ed.), vol. 3383 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 482–491. [98](#), [100](#)
-

- [Hol06] HOLTEN D.: Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *Visualization and Computer Graphics, IEEE Transactions on* 12, 5 (2006), 741–748. [20](#)
- [HR02] HINTON G. E., ROWEIS S. T.: Stochastic neighbor embedding. In *Advances in neural information processing systems* (2002), pp. 833–840. [123](#)
- [HR07] HEER J., ROBERTSON G.: Animated transitions in statistical data graphics. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (Nov 2007), 1240–1247. [23](#)
- [HS11] HAUNERT J.-H., SERING L.: Drawing road networks with focus regions. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2555–2562. [105](#)
- [HSCW13] HADLAK S., SCHUMANN H., CAP C. H., WOLLENBERG T.: Supporting the visual analysis of dynamic networks by clustering associated temporal attributes. *Visualization and Computer Graphics, IEEE Transactions on* 19, 12 (2013), 2267–2276. [95](#), [96](#), [123](#)
- [HSS11] HADLAK S., SCHULZ H.-J., SCHUMANN H.: In situ exploration of large dynamic networks. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2334–2343. [45](#), [106](#)
- [Hub85] HUBER P. J.: Projection pursuit. *The annals of Statistics* (1985), 435–475. [14](#)
- [IL08] INFELD D., LI F.: Integrating micro-generation into distribution systems; a review of recent research. In *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE* (july 2008), pp. 1–4. [88](#)
- [Imh75] IMHOF E.: Positioning names on maps. *The American Cartographer* 2, 2 (1975), 128–144. [103](#)
- [Ins85] INSELBERG A.: The plane with parallel coordinates. *The Visual Computer* 1, 2 (1985), 69–91. [16](#)
- [JD88] JAIN A. K., DUBES R. C.: *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988. [126](#)
- [JKKS12] JUSUFI I., KLUKAS C., KERREN A., SCHREIBER F.: Guiding the interactive exploration of metabolic pathway interconnections. *Information Visualization* 11, 2 (2012), 136–150. [41](#), [45](#)
- [JPC*11] JOIA P., PAULOVIH F., COIMBRA D., CUMINATO J., NONATO L.: Local affine multidimensional projection. *Visualization and Computer Graphics, IEEE Transactions on* 17, 12 (2011), 2563–2571. [81](#), [123](#), [124](#)

-
- [JSMK14] JANETZKO H., STOFFEL F., MITTELSTÄDT S., KEIM D. A.: Anomaly detection for visual analytics of power consumption data. *Computer & Graphics* 38 (2014), 27–37. [95](#)
- [Jul03] JULISCH K.: Clustering intrusion detection alarms to support root cause analysis. *ACM Trans. Inf. Syst. Secur.* 6, 4 (Nov. 2003), 443–471. [92](#)
- [KAF*08] KEIM D., ANDRIENKO G., FEKETE J.-D., GÖRG C., KOHLHAMMER J., MELANÇON G.: Visual analytics: Definition, process, and challenges. In *Information Visualization*, Kerren A., Stasko J., Fekete J.-D., North C., (Eds.), vol. 4950 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 154–175. [26](#), [27](#)
- [KAK95] KEIM D. A., ANKERST M., KRIEGEL H.-P.: Recursive pattern: A technique for visualizing very large amounts of data. In *Proceedings of the 6th Conference on Visualization '95* (Washington, DC, USA, 1995), VIS '95, IEEE Computer Society, p. 279. [16](#)
- [KCJ*10] KARNICK P., CLINE D., JESCHKE S., RAZDAN A., WONKA P.: Route visualization using detail lenses. *IEEE Transactions on Visualization and Computer Graphics* 16 (March 2010), 235–247. [41](#)
- [KDRK11] KRÄMER M., DUMMER M., RUPPERT T., KOHLHAMMER J.: Tackling uncertainty in combined visualizations of underground information and 3d city models. In *GeoViz Hamburg 2011 Workshop* (Hamburg, Germany, 2011), HafenCity University, Hamburg and International Cartographic Association (ICA): Commission on GeoVisualization, p. 2. [173](#)
- [KE05] KOHLHAMMER J., ENCARNAÇÃO J. L.: *Knowledge Representation for Decision-Centered Visualization*. GCA-Verlag, 2005. [94](#)
- [Kei00] KEIM D.: Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics* 6, 1 (2000), 59–78. [13](#), [15](#), [27](#), [116](#), [127](#)
- [KK89] KAMADA T., KAWAI S.: An algorithm for drawing general undirected graphs. *Information processing letters* 31, 1 (1989), 7–15. [37](#)
- [KK96] KEIM D., KRIEGEL H.-P.: Visualization techniques for mining large databases: a comparison. *Knowledge and Data Engineering, IEEE Transactions on* 8, 6 (1996), 923–938. [16](#)
- [KKC14] KERRACHER N., KENNEDY J., CHALMERS K.: The design space of temporal graph visualisation. In *EuroVis - Short Papers* (2014), Elmqvist N., Hlawitschka M., Kennedy J., (Eds.), Eurographics Association. [96](#)
-

- [KMH01] KOSARA R., MIKSCH S., HAUSER H.: Semantic depth of field. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 97–. [23](#)
- [KMS*08] KEIM D. A., MANSMANN F., SCHNEIDEWIND J., THOMAS J., ZIEGLER H.: Visual analytics: Scope and challenges. In *Visual Data Mining*, Simoff S. J., Böhlen M. H., Mazeika A., (Eds.), vol. 4404 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 76–90. [9](#), [25](#), [26](#), [27](#)
- [Kob12] KOBOUROV S. G.: Spring embedders and force directed graph drawing algorithms. *CoRR abs/1201.3011* (2012). [70](#)
- [Kor03] KOREN Y.: On Spectral Graph Drawing. In *Computing and Combinatorics*, Warnow T., Zhu B., (Eds.), vol. 2697 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003, pp. 496–508. [37](#)
- [Kos07] KOSSLYN S.: Remembering images. *Memory and Mind: A Festschrift for Gordon H. Bower* (2007), 93–110. [44](#)
- [KPSN04] KEIM D., PANSE C., SIPS M., NORTH S.: Visual data mining in large geospatial point sets. *Computer Graphics and Applications, IEEE* 24, 5 (Sept 2004), 36–44. [25](#)
- [Kru64] KRUSKAL J.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika* 29, 1 (1964), 1–27. [123](#)
- [KSH07] KYRIAKIDES E., STAHLHUT J., HEYDT G.: A next generation alarm processing algorithm incorporating recommendations and decisions on wide area control. In *Power Engineering Society General Meeting, 2007. IEEE* (june 2007), pp. 1 –5. [89](#), [92](#), [110](#)
- [KSO02] KLUMP R., SCHOOLEY D., OVERBYE T.: An advanced visualization platform for real-time power system operations. In *Proc. of the 14th Power Systems Computation Conference* (2002). [94](#), [95](#), [100](#)
- [KW01] KAUFMANN M., WAGNER D.: *Drawing Graphs: Methods and Models*, vol. 2025. Springer, 2001. [90](#)
- [Lau05] LAUFENBERG M.: Integration of large-scale visualization systems into a control center. In *Power Engineering Society General Meeting* (June 2005), vol. 3, IEEE, pp. 2712 – 2714. [94](#)
- [LCL06] LUO M. R., CUI G., LI C.: Uniform colour spaces based on ciecam02 colour appearance model. *Color Research & Application* 31, 4 (2006), 320–330. [137](#), [143](#)

-
- [LLY06] LEE Y.-Y., LIN C.-C., YEN H.-C.: Mental map preserving graph drawing using simulated annealing. In *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation - Volume 60* (Darlinghurst, Australia, Australia, 2006), APVis '06, Australian Computer Society, Inc., pp. 179–188. [40](#)
- [LMvW10] LI J., MARTENS J.-B., VAN WIJK J. J.: Judging correlation from scatterplots and parallel coordinate plots. *Information Visualization* 9, 1 (2010), 13–30. [16](#)
- [Lor04] LORENSEN B.: On the death of visualization. In *Position Papers NIH/NSF Proc. Fall 2004 Workshop Visualization Research Challenges* (2004), vol. 1. [21](#)
- [LPP*06] LEE B., PLAISANT C., PARR C. S., FEKETE J.-D., HENRY N.: Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI workshop on beyond time and errors* (New York, USA, 2006), BELIV '06, ACM, pp. 1–5. [33](#)
- [LQS*10] LI F., QIAO W., SUN H., WAN H., WANG J., XIA Y., XU Z., ZHANG P.: Smart transmission grid: Vision and framework. *Smart Grid, IEEE Transactions on* 1, 2 (sept. 2010), 168–177. [93](#), [94](#)
- [LRHS14] LUBOSCHIK M., RYBACKI S., HAACK F., SCHULZ H.-J.: Supporting the integrated visual analysis of input parameters and simulation trajectories. *Computers & Graphics* 39, 0 (2014), 37–47. [153](#)
- [LRP95] LAMPING J., RAO R., PIROLLO P.: A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1995), CHI '95, ACM Press/Addison-Wesley Publishing Co., pp. 401–408. [36](#), [38](#)
- [LS10] LIU Z., TASKO J.: Mental models, visual reasoning and interaction in information visualization: A top-down perspective. *IEEE Transactions on Visualization and Computer Graphics* 16, 6 (Nov. 2010), 999–1008. [11](#), [12](#)
- [LT11] LI X., TATE J.: Estimating and visualizing the impact of wind forecast errors on system operations. In *Power and Energy Society General Meeting, 2011 IEEE* (july 2011), pp. 1–7. [93](#)
- [LV10] LEE J. A., VERLEYSEN M.: Unsupervised dimensionality reduction: Overview and recent advances. In *The International Joint Conference on Neural Networks* (2010), pp. 1–8. [123](#)
- [Mac42] MACADAM D. L.: Visual sensitivities to color differences in daylight. *Journal of the Optical Society of America* 32, 5 (May 1942), 247–273. [145](#)
-

- [MBS*14] MITTELSTÄDT S., BERNARD J., SCHRECK T., STEIGER M., KOHLHAMMER J., KEIM D. A.: Revisiting Perceptually Optimized Color Mapping for High-Dimensional Data Analysis. In *Proceedings of the Eurographics Conference on Visualization (EuroVis 2014, Short Paper)* (2014). [136](#), [150](#), [172](#)
- [MCH*09] MOSCOVICH T., CHEVALIER F., HENRY N., PIETRIGA E., FEKETE J.-D.: Topology-aware navigation in large networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2009), CHI '09, ACM, pp. 2319–2328. [39](#), [41](#)
- [MCO04] MELIOPOULOS A., COKKINIDES G., OVERBYE T.: Component monitoring and dynamic loading visualization from real time power flow model data. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on* (2004), pp. 6 pp.–. [95](#)
- [MELS95] MISUE K., EADES P., LAI W., SUGIYAMA K.: Layout adjustment and the mental map. *Journal of Visual Languages & Computing* 6, 2 (1995), 183 – 210. [39](#)
- [MFH*02] MORONEY N., FAIRCHILD M. D., HUNT R. W., LI C., LUO M. R., NEWMAN T.: The CIECAM02 color appearance model. In *Proceedings of the Color and Imaging Conference* (2002), vol. 2002/1, pp. 23–27. [137](#)
- [MG07] MERRICK D., GUDMUNDSSON J.: Path simplification for metro map layout. In *Graph Drawing*, Kaufmann M., Wagner D., (Eds.), vol. 4372 of *Lecture Notes in Computer Science*. Springer, Heidelberg, 2007, pp. 258–269. [100](#), [102](#), [103](#)
- [MGJH11] MATKOVIC K., GRACANIN D., JELOVIC M., HAUSER H.: Interactive visual analysis supporting design, tuning, and optimization of diesel engine injection. In *Proceedings of IEEE Visualization* (2011), vol. 2011, p. 3. [153](#)
- [MHS07] MACKINLAY J., HANRAHAN P., STOLTE C.: Show me: Automatic presentation for visual analysis. *Visualization and Computer Graphics, IEEE Transactions on* 13, 6 (nov.-dec. 2007), 1137–1144. [94](#)
- [MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless Deformations Based on Shape Matching. *ACM Trans. Graph.* 24, 3 (July 2005), 471–478. [70](#), [79](#)
- [Mie14] MIETTINEN K.: Survey of methods to visualize alternatives in multiple criteria decision making problems. *OR Spectrum* 36, 1 (2014), 3–37. [153](#)
- [MK08] MAY T., KOHLHAMMER J.: Towards closing the analysis gap: Visual generation of decision supporting schemes from raw data. *Computer Graphics Forum* 27, 3 (2008), 911–918. [17](#), [18](#)

-
- [MKN*07] MANSMANN F., KEIM D., NORTH S., REXROAD B., SHELEHEDA D.: Visual analysis of network traffic for resource planning, interactive monitoring, and interpretation of security threats. *Visualization and Computer Graphics, IEEE Transactions on* 13, 6 (2007), 1105–1112. [19](#)
- [MMKN08] MCLACHLAN P., MUNZNER T., KOUTSOFIOS E., NORTH S.: Liverac: interactive visual exploration of system management time-series data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2008), CHI '08, ACM, pp. 1483–1492. [93](#)
- [Mor96] MORRISON A.: Public transport maps in western european cities. *The Cartographic Journal* 33, 2 (1996), 93–110. [97](#)
- [MPWG12] MARRIOTT K., PURCHASE H., WYBROW M., GONCU C.: Memorability of Visual Features in Network Diagrams. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (dec. 2012), 2477–2485. [39](#), [40](#)
- [MSDK12] MAY T., STEIGER M., DAVEY J., KOHLHAMMER J.: Using Signposts for Navigation in Large Graphs. *Computer Graphics Forum* 31, 3 pt. 2 (2012), 985–994. [3](#), [30](#), [108](#), [172](#)
- [MSS*13] MITTELSTÄDT S., SPRETKE D., SACHA D., KEIM D. A., HEYDER B., KOPP J.: Visual analytics for critical infrastructures. In *Security in Critical Infrastructures Today, Proc. of Int. ETG-Congress; Symposium 1* (2013), pp. 1–8. [106](#), [107](#), [109](#)
- [MTF05] MARQUES A., TARANTO G., FALCAO D.: A knowledge-based system for supervision and control of regional voltage profile and security. *Power Systems, IEEE Transactions on* 20, 1 (feb. 2005), 400 – 407. [92](#)
- [Mun98] MUNZNER T.: Exploring large graphs in 3d hyperbolic space. *Computer Graphics and Applications, IEEE* 18, 4 (1998), 18–23. [36](#)
- [Nes04] NESBITT K.: Getting to more abstract places using the metro map metaphor. In *Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on* (2004), pp. 488–493. [107](#)
- [NMPR07] NEUMAYER R., MAYER R., PÖLZLBAUER G., RAUBER A.: The metro visualisation of component planes for self-organising maps. In *International Joint Conference on Neural Networks* (2007), pp. 2788–2793. [107](#)
- [Nöl05] NÖLLENBURG M.: *Automated drawing of metro maps*. Master's thesis, Universität Karlsruhe, Fakultät für Informatik, 2005. [97](#)
- [Nor96] NORTH S. C.: Incremental layout in dynadag. In *Proceedings of the Symposium on Graph Drawing* (London, UK, 1996), GD '95, Springer, pp. 409–418. [38](#), [67](#)
-

- [Nor06] NORTH C.: Toward measuring visualization insight. *Computer Graphics and Applications, IEEE* 26, 3 (May 2006), 6–9. [2](#)
- [NS00] NORTH C., SHNEIDERMAN B.: Snap-together visualization: A user interface for coordinating visualizations via relational schemata. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (New York, NY, USA, 2000), AVI '00, ACM, pp. 128–135. [14](#), [22](#)
- [NSBK14] NAZEMI K., STEIGER M., BURKHARDT D., KOHLHAMMER J.: *Handbook of Research on Advanced ICT Integration for Governance and Policy Modeling*. IGI Global, Hershey, Pennsylvania, 2014, book chapter Information Visualization and Policy Modeling. [7](#), [171](#)
- [NW06] NÖLLENBURG M., WOLFF A.: A mixed-integer program for drawing high-quality metro maps. In *Graph Drawing*, Healy P., Nikolov N., (Eds.), vol. 3843 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 321–333. [101](#), [122](#), [128](#)
- [NW11] NÖLLENBURG M., WOLFF A.: Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Transactions on Vis. and CG.* 17, 5 (2011), 626–641. [99](#), [101](#)
- [OMWC05] OVERBYE T., MELIOPOULOS A., WIEGMANN D., COKKINIDES G.: *Visualization of power systems and components*. Tech. rep., Power Systems Engineering Research Center(PSERC), 2005. [93](#), [94](#), [95](#), [105](#), [106](#)
- [Osa01] OSAWA N.: A multiple-focus graph browsing technique using heat models and force-directed layout. In *Proceedings of the Fifth International Conference on Information Visualisation* (Washington, DC, USA, 2001), IV '01, IEEE Computer Society, pp. 277–277. [40](#)
- [Ove05] OVENDEN M.: *Metro maps of the world*. Capital Transport, 2005. [97](#)
- [OW01] OVERBYE T., WEBER J.: Visualizing the electric grid. *Spectrum, IEEE* 38, 2 (Feb 2001), 52–58. [95](#)
- [OWL99] OVERBYE T. J., WEBER J. D., LAUFENBERG M. J.: Visualization of flows and transfer capability in electric networks. In *Proceedings of the 13th Power Systems Computation Conference* (June 1999), pp. 420–426. [95](#), [105](#)
- [OWRS03] OVERBYE T., WIEGMANN D., RICH A., SUN Y.: Human factors aspects of power system voltage contour visualizations. *Power Systems, IEEE Transactions on* 18, 1 (2003), 76–82. [95](#)
- [PBK10] PIRINGER H., BERGER W., KRASSER J.: HyperMoVal: Interactive Visual Validation of Regression Models for Real-Time Simulation. In *Computer Graphics Forum* (2010), vol. 29/3, Wiley Online Library, pp. 983–992. [153](#)

-
- [PHG07] PURCHASE H. C., HOGGAN E., GÖRG C.: How Important Is the "Mental Map"? - An Empirical Investigation of a Dynamic Graph Layout Algorithm. In *Graph Drawing*, Kaufmann M., Wagner D., (Eds.), vol. 4372 of *LNCS*. Springer, 2007, pp. 184–195. [39](#)
- [PKS10] PARIKH P., KANABAR M., SIDHU T.: Opportunities and challenges of wireless communication technologies for smart grid applications. In *Power and Energy Society General Meeting, 2010 IEEE* (2010), pp. 1–7. [111](#)
- [PS08] PURCHASE H. C., SAMRA A.: Extremes are better: Investigating mental map preservation in dynamic graphs. In *Proceedings of the 5th international conference on Diagrammatic Representation and Inference* (2008), Springer, pp. 60–73. [39](#), [40](#)
- [PV06] PAPADOPOULOS C., VOGLIS C.: Drawing graphs using modular decomposition. In *Graph Drawing*, Healy P., Nikolov N., (Eds.), vol. 3843 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 343–354. [38](#)
- [RC94] RAO R., CARD S. K.: The table lens: Merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1994), CHI '94, ACM, pp. 318–322. [23](#)
- [RH04] ROONEY J. J., HEUVEL L. N. V.: Root cause analysis for beginners. *Quality progress* 37, 7 (2004), 45–56. [92](#)
- [Rhe00] RHEINGANS P.: Task-based color scale design. In *28th AIPR Workshop: 3D Visualization for Data Exploration and Decision Making* (2000), International Society for Optics and Photonics. [136](#)
- [RI04] REILLY D. F., INKPEN K. M.: Map morphing: Making sense of incongruent maps. In *Proceedings of Graphics Interface 2004* (2004), Canadian Human-Computer Communications Society, pp. 231–238. [105](#)
- [RI07] REILLY D. F., INKPEN K. M.: White rooms and morphing don't mix: Setting and the evaluation of visualization techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2007), CHI '07, ACM, pp. 111–120. [105](#)
- [RM93] ROBERTSON G. G., MACKINLAY J. D.: The document lens. In *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 1993), UIST '93, ACM, pp. 101–108. [23](#)
- [RMC91] ROBERTSON G. G., MACKINLAY J. D., CARD S. K.: Cone trees: Animated 3d visualizations of hierarchical information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1991), CHI '91, ACM, pp. 189–194. [19](#)
-

- [RNL*13] ROBERTS M. J., NEWTON E. J., LAGATTOLLA F. D., HUGHES S., HASLER M. C.: Objective versus subjective measures of paris metro map usability: Investigating traditional octolinear versus all-curves schematics. *International Journal of Human-Computer Studies* 71, 3 (2013), 363–386. [101](#), [103](#)
- [RO86] ROBERTSON P. K., O’CALLAGHAN J. F.: The generation of color sequences for univariate and bivariate mapping. *Computer Graphics and Applications, IEEE* 6, 2 (1986), 24–32. [136](#), [137](#)
- [Rob12] ROBERTS M. J.: *Underground maps unravelled: Explorations in information design*. Maxwell J. Roberts, 2012. [100](#), [103](#)
- [RT81] REINGOLD E. M., TILFORD J.: Tidier drawings of trees. *Software Engineering, IEEE Transactions on SE-7*, 2 (1981), 223–228. [19](#)
- [RTB96] ROGOWITZ B., TREINISH L., BRYSON S.: How not to lie with visualization. *Computers in Physics* 10, 3 (1996). [136](#)
- [SA99] SIMULA O., ALHONIEMI E.: Som based analysis of pulping process data. In *Engineering Applications of Bio-Inspired Artificial Neural Networks*. Springer, 1999, pp. 567–577. [137](#), [144](#), [149](#)
- [SB92] SARKAR M., BROWN M. H.: Graphical fisheye views of graphs. In *Proc. of the 27th International Conference on Human Factors in Computing Systems (CHI)* (New York, NY, USA, 1992), CHI ’92, ACM, pp. 83–91. [44](#)
- [SBM08] SIMOFF S. J., BÖHLEN M. H., MAZEIKA A.: Visual data mining: An introduction and overview. In *Visual Data Mining*, Simoff S. J., Böhlen M. H., Mazeika A., (Eds.), vol. 4404 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 1–12. [25](#)
- [SBM*14] STEIGER M., BERNARD J., MITTELSTÄDT S., LÜCKE-TIEKE H., KEIM D., MAY T., KOHLHAMMER J.: Visual analysis of time-series similarities for anomaly detection in sensor networks. *Computer Graphics Forum* 33, 3 (2014), 401–410. [4](#), [88](#), [137](#), [171](#)
- [SBM*15] STEIGER M., BERNARD J., MITTELSTÄDT S., HUTTER M., KEIM D., THUM S., KOHLHAMMER J.: Explorative analysis of 2d color maps. [88](#), [171](#)
- [SBMK14] STEIGER M., BERNARD J., MAY T., KOHLHAMMER J.: A Survey of Direction-Preserving Layout Strategies. In *Proceedings of the 30th Spring Conference on Computer Graphics* (New York, NY, USA, 2014), SCCG ’14, ACM, pp. 50 – 57. [88](#), [171](#)
- [SBSK15] STEIGER M., BERNARD J., SCHADER P., KOHLHAMMER J.: Visual analysis of relations in attributed time-series data. In *EuroVA 2015* (2015), European Association for Computer Graphics (Eurographics), Eurographics Association, Goslar. [172](#)

-
- [SBvLK09] SCHRECK T., BERNARD J., VON LANDESBERGER T., KOHLHAMMER J.: Visual cluster analysis of trajectory data with interactive kohonen maps. *Information Visualization* 8, 1 (2009), 14–29. [18](#)
- [SC07] SALVADOR S., CHAN P.: Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis* 11, 5 (2007), 561–580. [123](#)
- [SFK13] STOFFEL F., FISCHER F., KEIM D. A.: Finding anomalies in time-series using visual correlation for interactive root cause analysis. In *Proceedings of the Tenth Workshop on Visualization for Cyber Security* (New York, USA, 2013), VizSec '13, ACM, pp. 65–72. [94](#)
- [SGBBT06] SHAHAR Y., GOREN-BAR D., BOAZ D., TAHAN G.: Distributed, intelligent, interactive visualization and exploration of time-oriented clinical data and their abstractions. *Artif. Intell. Med.* 38, 2 (Oct. 2006), 115–135. [93](#)
- [SGH12] SHAHAF D., GUESTRIN C., HORVITZ E.: Metro maps of science. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2012), KDD '12, ACM, pp. 1122–1130. [107](#)
- [Shn96] SHNEIDERMAN B.: The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on* (1996), IEEE, pp. 336–343. [13](#), [27](#), [42](#)
- [SHS*14] STEIGER M., HUTTER M., SCHADER P., KOHLHAMMER J., KUIJPER A.: Exploring simulation in sensor network models. In *VMV 2014* (2014), European Association for Computer Graphics (Eurographics), Eurographics Association, Goslar, pp. 135–142. [4](#), [88](#), [171](#)
- [Sii00] SIIRTOLA H.: Direct manipulation of parallel coordinates. In *Information Visualization, 2000. Proceedings. IEEE International Conference on* (2000), pp. 373–378. [16](#)
- [Sii07] SIIRTOLA H.: *Interactive Visualization of Multidimensional Data*. Dissertations in Interactive Technology, Number 7, University of Tampere, 2007. [7](#)
- [SKRK13] STEIGER M., KRÄMER M., RUPPERT T., KOHLHAMMER J.: Visualizing uncertain underground information for urban management. In *Proceedings of the IADIS International Conference Computer Graphics, Visualization, Computer Vision and Image Processing* (2013), International Association for Development of the Information Society (IADIS), IADIS Press, pp. 75–82. [173](#)
- [SLH*11] SHI L., LIAO Q., HE Y., LI R., STRIEGEL A., SU Z.: Save: Sensor anomaly visualization engine. In *IEEE Conference on Visual Analytics Science and Technology (VAST)* (2011), pp. 201–210. [95](#)
-

- [SLN05] SARAIYA P., LEE P., NORTH C.: Visualization of graphs with associated time-series data. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on* (2005), pp. 225–232. [95](#)
- [SLTM*13] STEIGER M., LÜCKE-TIEKE H., MAY T., KUIJPER A., KOHLHAMMER J.: Using layout stitching to create deterministic local graph layouts. In *WSCG'13* (June 2013), pp. 1–9. [4](#), [30](#), [82](#), [172](#)
- [SLTM*14] STEIGER M., LÜCKE-TIEKE H., MAY T., KUIJPER A., KOHLHAMMER J.: Deterministic local layouts through high-dimensional layout stitching. In *Human-Computer Interaction. Theories, Methods, and Tools*, Kurosu M., (Ed.), vol. 8510 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 643–651. [4](#), [30](#), [171](#)
- [SMDK13a] STEIGER M., MAY T., DAVEY J., KOHLHAMMER J.: Smart grid monitoring through visual analysis. In *Innovative Smart Grid Technologies Europe (ISGT EUROPE), 2013 4th IEEE/PES* (2013), pp. 1–5. [4](#), [88](#), [172](#)
- [SMDK13b] STEIGER M., MAY T., DAVEY J., KOHLHAMMER J.: Visual analysis of expert systems for smart grid monitoring. In *EuroVA 2013* (2013), European Association for Computer Graphics (Eurographics), Eurographics Association, Goslar, pp. 43–47. [4](#), [88](#), [172](#)
- [Smi78] SMITH A. R.: Color gamut transform pairs. In *ACM Siggraph Computer Graphics* (1978), vol. 12/3, ACM, pp. 12–19. [143](#)
- [SMK13] STEIGER M., MAY T., KOHLHAMMER J.: Stable incremental layouts for dynamic graph visualizations. In *Proceedings of the IADIS International Conference Computer Graphics, Visualization, Computer Vision and Image Processing* (2013), International Association for Development of the Information Society (IADIS), IADIS Press, pp. 91–98. [3](#), [30](#), [171](#)
- [SMK14] STEIGER M., MAY T., KOHLHAMMER J.: Stable incremental layouts for dynamic graph visualizations. *International Journal on Computer Science and Information Systems* 8, 2 (2014), 12–26. [3](#), [30](#), [171](#)
- [SNLH09] SIPS M., NEUBERT B., LEWIS J. P., HANRAHAN P.: Selecting good views of high-dimensional data using class consistency. *Computer Graphics Forum* 28, 3 (2009), 831–838. [123](#), [124](#)
- [SO04] SUN Y., OVERBYE T.: Visualizations for power system contingency analysis data. *Power Systems, IEEE Transactions on* 19, 4 (2004), 1859–1866. [95](#), [105](#)
- [SP04] STAN SALVADOR, PHILIP CHAN: FastDTW: toward accurate dynamic time warping in linear time and space. In *KDD Workshop on Mining Temporal and Sequential Data* (2004), pp. 70–80. [159](#)

-
- [SP07] SCHRECK T., PANSE C.: A new metaphor for projection-based visual analysis and data exploration. *Proc. SPIE 6495* (2007). [129](#), [133](#)
- [SP08] SAFFREY P., PURCHASE H.: The "mental map" versus "static aesthetic" compromise in dynamic graphs: a user study. In *Proceedings of the ninth conference on Australasian user interface - Volume 76* (Darlinghurst, Australia, Australia, 2008), AUIC '08, Australian Computer Society, Inc., pp. 85–93. [40](#)
- [SPCJ09] SHNEIDERMAN B., PLAISANT C., COHEN M., JACOBS S.: *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 5th ed. Addison-Wesley Publishing Company, USA, 2009. [21](#)
- [SQW11] SHI J., QIAO Y., WANG H.: Visualizing inference process of a rule engine. In *Proceedings of the 2011 Visual Information Communication - International Symposium* (New York, NY, USA, 2011), VINCI '11, ACM, pp. 10:1–10:9. [94](#)
- [SR05] STOTT J. M., RODGERS P.: Automatic metro map design techniques. In *Proceedings of the 22nd International Cartographic Conference* (2005). [98](#), [101](#)
- [SRMOW11] STOTT J., RODGERS P., MARTÍNEZ-OVANDO J., WALKER S.: Automatic metro map layout using multicriteria optimization. *IEEE Transactions on Vis. and CG.* 17, 1 (2011), 101–114. [101](#)
- [SvLB10] SCHRECK T., VON LANDESBERGER T., BREMM S.: Techniques for precision-based visual analysis of projected data. *Information Visualization* 9, 3 (2010), 181–193. [137](#)
- [SW97] SPENCE I., WAINER H.: Who was playfair? *Chance* 10 (1997), 35–37. [7](#)
- [Tam87] TAMASSIA R.: On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing* 16, 3 (1987), 421–444. [100](#)
- [TAvHS06] TOMINSKI C., ABELLO J., VAN HAM F., SCHUMANN H.: Fisheye tree views and lenses for graph visualization. In *Proc. of the Conference on Information Visualization (IV)* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 17–24. [41](#)
- [TBET98] TOLLIS I. G., BATTISTA G. D., EADES P., TAMASSIA R.: *Graph Drawing: Algorithms for the Visualization of Graphs*, vol. 1. Prentice Hall, New York, 1998. [98](#)
- [TC05] THOMAS J. J., COOK K. A.: *Illuminating the path: The research and development agenda for visual analytics*. IEEE Computer Society Press, 2005. [2](#), [25](#)
-

- [TFS08] TOMINSKI C., FUCHS G., SCHUMANN H.: Task-driven color coding. In *Information Visualisation, 2008. IV'08. 12th International Conference* (2008), IEEE, pp. 373–380. [135](#)
- [TK07] TEKUŠOVÁ T., KOHLHAMMER J.: Applying animation to the visual analysis of financial time-dependent data. In *Information Visualization, 2007. IV '07. 11th International Conference* (July 2007), pp. 101–108. [23](#)
- [Tor52] TORGERSON W. S.: Multidimensional scaling: I. Theory and Method. *Psychometrika* 17, 4 (1952), 401–419. [81](#), [123](#)
- [TRFBMBM10] TORRALBA-RODRÍGUEZ F. J., FERNÁNDEZ-BREIS J. T., MARTÍNEZ-BÉJAR R., MONTAGUD V. B.: An incremental knowledge acquisition-based system for critical domains. *Expert Systems with Applications* 37, 4 (2010), 2838 – 2847. [93](#)
- [TS08] TOMINSKI C., SCHUMANN H.: Enhanced Interactive Spiral Display. In *Proceedings of the Annual SIGRAD Conference, Special Theme: Interactivity* (2008), Linköping University Electronic Press, pp. 53–56. [95](#)
- [TSS11] TEULING A. J., STÖCKLI R., SENEVIRATNE S. I.: Bivariate colour maps for visualizing climate data. *International Journal of Climatology* 31, 9 (2011), 1408–1412. [149](#)
- [Tuf97] TUFTE E. R.: *Visual explanations: images and quantities, evidence and narrative*, vol. 107. Graphics Press Cheshire, CT, 1997. [97](#)
- [US 08] US ENVIRONMENTAL PROTECTION AGENCY: EPANET, 2008. [159](#)
- [US09] UNGER A., SCHUMANN H.: Visual support for the understanding of simulation processes. In *Visualization Symposium, 2009. PacificVis '09. IEEE Pacific* (April 2009), pp. 57–64. [152](#)
- [vHP09] VAN HAM F., PERER A.: Search, Show Context, Expand on Demand: Supporting Large Graph Exploration with Degree-of-Interest. *IEEE Transactions on Visualization and Computer Graphics* 15 (2009), 953–960. [38](#), [41](#), [42](#), [44](#), [45](#), [47](#), [57](#)
- [vHP12] VAN HAM F., PERER A.: Integrating Querying and Browsing in Partial Graph Visualizations. *IBM Technical Report 12-01* (2012). [39](#), [41](#)
- [vLKS*11] VON LANDESBERGER T., KUIJPER A., SCHRECK T., KOHLHAMMER J., VAN WIJK J., FEKETE J.-D., FELLNER D. W.: Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges. *Computer Graphics Forum* (2011). [34](#)
- [vW05] VAN WIJK J. J.: The value of visualization. In *Visualization, 2005. VIS 05. IEEE* (Oct 2005), pp. 79–86. [2](#), [21](#)

-
- [VWVS99] VAN WIJK J. J., VAN SELOW E. R.: Cluster and calendar based visualization of time series data. In *Proceedings of the Symposium on Information Visualization* (Washington, DC, USA, 1999), INFOVIS, IEEE Computer Society, pp. 4–9. [94](#), [129](#)
- [War88] WARE C.: Color sequences for univariate maps: Theory, experiments and principles. *IEEE Computer Graphics and Applications* 8, 5 (1988). [136](#)
- [War12] WARE C.: *Information visualization: perception for design*. Elsevier, 2012. [11](#)
- [WB88] WARE C., BEATTY J. C.: Using color dimensions to display data dimensions. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 30, 2 (1988), 127–142. [136](#)
- [WBD00] WRIGHT H., BRODLIE K., DAVID T.: Navigating high-dimensional spaces to support design steering. In *Visualization 2000. Proceedings* (2000), pp. 291–296. [25](#), [151](#), [153](#)
- [WC11] WANG Y.-S., CHI M.-T.: Focus+context metro maps. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2528–2535. [105](#)
- [WF80] WAINER H., FRANCOLINI C. M.: An empirical inquiry concerning human understanding of two-variable color maps. *The American Statistician* 34, 2 (1980), 81–93. [135](#), [136](#)
- [WG11] WARD M. O., GUO Z.: Visual exploration of time-series data with shape space projections. In *Proceedings of the Eurographics Conference on Visualization* (2011), EuroVis, Eurographics Association, pp. 701–710. [95](#)
- [Wol07] WOLFF A.: Drawing subway maps: A survey. *Informatik - Forschung und Entwicklung* 22, 1 (2007), 23–44. [98](#), [100](#)
- [WRR03] WALL M., RECHTSTEINER A., ROCHA L.: Singular Value Decomposition and Principal Component Analysis. *A practical approach to microarray data analysis* (2003), 91–109. [71](#)
- [WSM*09] WONG P. C., SCHNEIDER K., MACKEY P., FOOTE H., CHIN G., GUTTROMSON R., THOMAS J.: A novel visualization technique for electric power grid analytics. *Visualization and Computer Graphics, IEEE Transactions on* 15, 3 (2009), 410–423. [95](#), [101](#), [102](#), [106](#)
- [WTAT06] WARE J. M., TAYLOR G. E., ANAND S., THOMAS N.: Automated production of schematic maps for mobile applications. *Transactions in GIS* 10, 1 (2006), 25–42. [102](#)
- [WTH*13] WU H.-Y., TAKAHASHI S., HIRONO D., ARIKAWA M., LIN C.-C., YEN H.-C.: Spatially efficient design of annotated metro maps. *CG. Forum* 32, 3pt3 (2013), 261–270. [104](#)
-

- [XGH06] XIAO L., GERTH J., HANRAHAN P.: Enhancing visual analysis of network traffic using a knowledge representation. In *Visual Analytics Science And Technology, 2006 IEEE Symposium On* (31 2006-nov. 2 2006), pp. 107–114. [94](#)
- [YCHZ12] YUAN X., CHE L., HU Y., ZHANG X.: Intelligent graph layout using many users' input. *Visualization and Computer Graphics, IEEE Transactions on* 18, 12 (2012), 2699–2708. [38](#)
- [ZFH08] ZHAO J., FORER P., HARVEY A. S.: Activities, ringmaps and geovisualization of large human movement fields. *Information Visualization* 7, 3-4 (2008), 198–209. [95](#)
- [Zha96] ZHANG J.: A representational analysis of relational information displays. *International Journal of Human-Computer Studies* 45, 1 (1996), 59–74. [8](#)
- [ZNK07] ZIEGLER H., NIETZSCHMANN T., KEIM D.: Visual exploration and discovery of atypical behavior in financial time series data using two-dimensional colormaps. In *Information Visualization, 2007. IV '07. 11th International Conference* (2007), pp. 308–315. [127](#), [137](#)